Thermal and IR Drop Analysis
Using Convolutional Encoder-Decoder Networks

Vidya A. Chhabria[1], Vipul Ahuja[2], Ashwath Prabhu[2], Nikhil Patil[2],
Palkesh Jain[2], and Sachin S. Sapatnekar[1]
[1]University of Minnesota, USA; [2]Qualcomm Technologies Inc., India

Abstract

Computationally expensive temperature and power grid analyses are required during the design cycle to guide IC design. This paper employs encoder-decoder based generative (EDGe) networks to map these analyses to fast and accurate image-to-image and sequence-to-sequence translation tasks. The network takes a power map as input and outputs the temperature or IR drop map. We propose two networks: (i) ThermEDGe: a static and dynamic full-chip temperature estimator and (ii) IREDGe: a full-chip static IR drop predictor based on input power, power grid distribution, and power pad distribution patterns. The models are design-independent and must be trained just once for a particular technology and packaging solution. ThermEDGe and IREDGe are demonstrated to rapidly predict on-chip temperature and IR drop contours in milliseconds (in contrast with commercial tools that require several hours or more) and provide an average error of 0.6% and 0.008% respectively.

ACM Reference Format:
Vidya A. Chhabria[1], Vipul Ahuja[2], Ashwath Prabhu[2], Nikhil Patil[2], and Palkesh Jain[2], and Sachin S. Sapatnekar[1]. 2021. Thermal and IR Drop Analysis Using Convolutional Encoder-Decoder Networks. In *26th Asia and South Pacific Design Automation Conference (ASPDAC '21), January 18–21, 2021, Tokyo, Japan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3394885.3431583

1 Introduction

One of the major challenges faced by an advanced-technology node IC designer is the overhead of large run-times of analysis tools. Fast and accurate analysis tools that aid quick design turn-around are particularly important for two critical, time-consuming simulations that are performed several times during the design cycle:

- *Thermal analysis* which checks the feasibility of a placement/floorplan solution by computing on-chip temperature distributions in order to check for temperature hot spots.
- *IR drop analysis* in power distribution networks (PDNs), which diagnoses the goodness of the PDN by determining voltage (IR) drops from the power pads to the gates.

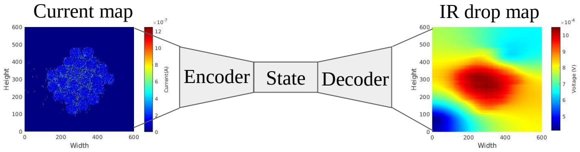

Figure 1: Image-to-image translation using EDGe network.

The underlying computational engines that form the crux of both analyses are similar: both simulate networks of conductances and current/voltage sources by solving a large system of equations of the form $G\mathbf{V} = \mathbf{J}$ [1, 2] with millions to billions of variables. In modern industry designs, a single full-chip temperature or IR drop simulation can take hours to several hours. Accelerating these analyses opens the door to optimizations in the design cycle that iteratively invoke these engines under the hood.

Prior acceleration methods such as [3] techniques trade off accuracy for speed by using a coarser finite element granularity. However, the advent of machine learning (ML) has presented fast and more accurate solutions by to these problems [4–11] by using data from simulations of systems of millions of nodes. For thermal analysis, existing ML-based solutions largely focus on coarser system-level thermal modeling [4–6, 10]. The work in [11] predicts temperature at a finer granularity, but uses coarse temperature estimates as an input. The works in [6, 10] use post silicon performance metrics as input to predict full-chip temperature and are not suitable for predicting temperature during the design process. For PDN analysis, the works in [7, 8] address incremental analysis, and are not intended for full-chip estimation. The work in [9] proposes a convolutional neural network (CNN)-based implementation for full-chip IR drop prediction, using cell-level power maps as features. However, it assumes similar resistance from each cell to the power pads, which may not be valid for practical power grids with irregular grid density. The ML techniques in [9, 11] both divide the chip into regions (*tiles*), and the CNNs operate on each tile and its near neighbors. Selecting an appropriate tile and window size is nontrivial – small windows could violate the principle of locality [12], causing inaccuracies, while large windows could result in large models with significant runtimes for training and inference. Our approach bypasses window size selection by providing the entire power map as input, allowing ML to learn the window size for accurate estimation of temperature and IR drop.

We translate static analysis problems to an image-to-image translation task and dynamic analysis problems to video-to-video translation. The inputs to the thermal analysis problem are the power/current distributions and the outputs are the temperature contours, while the inputs for the PDN analysis task are power distributions, PDN density maps, and power bump pattern and the outputs are the IR drop contours. The predicted output temperature and IR drop contours are at the accuracy level of a million-node

ASPDAC '21, January 18–21, 2021, Tokyo, Japan

ACM ISBN 978-1-4503-7999-1/21/01…$15.00
https://doi.org/10.1145/3394885.3431583

ground truth simulation. For static analysis, we employ fully convolutional (FC) EDGe networks for rapid and accurate thermal and IR drop analysis. FC EDGe networks have proven to be very successful with image-related problems with 2-D spatially distributed data [13–16] when compared to other networks that operate without spatial correlation awareness. For transient analysis, we use long-short-term-memory (LSTM) based networks that maintain memory of analyses at prior time steps.

Based on these concepts, this work proposes two novel ML-based analyzers: **ThermEDGe** for both *full-chip* static and transient thermal analysis, and **IREDGe** for *full-chip* static IR drop estimation. The fast inference times of ThermEDGe and IREDGe enable **full-chip** thermal and IR drop analysis in milliseconds, as opposed to several hours using commercial tools. We obtain average error of 0.6% and 0.008% for ThermEDGe and IREDGe, respectively, over a range of testcases. Our software has been released on GitHub [17].

Fig. 1 shows a general structure of an EDGe network. It consists of two parts: (i) the encoder/downsampling path, which captures global features of the 2-D distributions of power dissipation, and produces a low-dimensional state space and (ii) the decoder/upsampling path, which transforms the state space into the required detailed outputs (temperature or IR drop contours). The EDGe network is well-suited for PDN/thermal analyses because:
(a) The convolutional nature of the encoder *captures the dependence of both problems on the spatial distributions of power.* Unlike CNNs, EDGe networks contain a decoder which acts as a generator to convert the extracted power and PDN density features into accurate high-dimensional full-chip temperature and IR drop contours.
(b) The trained EDGe network model for static analysis is *design-independent*: it only stores the weights of the convolutional kernel, and the same filter can be applied to *any* chip of any size for a given technology and packaging solution. The selection of the network topology (convolution filter size, number of convolution layers) is related to the expected sizes of the hotspots rather than the size of the chip: these sizes are generally similar for a given application domain, technology, and packaging choice.
(c) Unlike prior methods [9] that operate tile-by-tile, where finding the right tile and window size for accurate analysis is challenging, *the choice of window size is treated as an ML hyperparameter tuning problem* to decide the amount of input spatial information.

2 EDGe Network for Analysis

2.1 Problem Formulations

This section presents the ML-based framework for ThermEDGe and IREDGe. The first step is to extract an appropriate set of features from a standard design-flow environment. The layout database provides the locations of each instance and block in the layout, as outlined in Fig. 2(a). This may be combined with information from a power analysis tool (Fig. 2(b)) that is used to build a 2-D spatial power map over the die area.

For thermal analysis using ThermEDGe, both the inputs and outputs are images for the static case, and a sequence of images for the transient case. Each input image shows a 2-D die power distribution (static) image, and each output image is a temperature map across the die (Fig. 2). For static PDN analysis, the output is an IR drop map across the full chip. However, in addition to the 2-D power distributions, IREDGe has two other inputs:

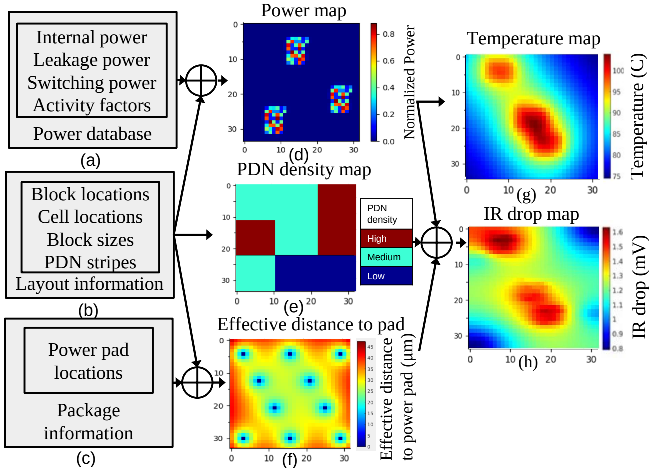

Figure 2: Data representation: Mapping PDN and thermal analysis problems into image-to-image translations tasks.

(i) *A PDN density map*: This feature is generated by extracting the average PDN pitch in each region of the chip. For example, when used in conjunction with the PDN styles in [18, 19], where the chip uses regionwise uniform PDNs, the average PDN density in each region, across all metal layers, is provided as an input (Fig. 2(e)).
(ii) *An effective distance to power pad*: This feature represents the equivalent distance from an instance to all power pads in the package. We compute the effective distance of each instance, d_e, to N power pads as $d_e^{-1} = d_1^{-1} + d_2^{-1} + \ldots + d_N^{-1}$ where d_i is the distance of the i^{th} power pad from the instance. Intuitively, the effective distance metric and the PDN density map together, represent the equivalent resistance between the instance and the pad. The equivalent resistance is a parallel combination of each path from the instance to the pad. We use distance to each pad as a proxy for the resistance. Fig. 2(f) shows a typical "checkerboard" power pad layout for flip-chip packages [20, 21].

Temperature depends on the ability of the package and system to conduct heat to the ambient, and IR drop depends on off-chip (e.g., package) parasitics. In this work, our focus is strictly on-chip, and both ThermEDGe and IR-EDGe are trained for fixed models of a given technology, package, and system.

Next, we map these problems to standard ML networks:

- For static analysis, the problem formulations require a translation from an input power image to an output image, both corresponding to contour maps over the same die area, and we employ a *U-Net-based EDGe network* [14].
- The dynamic analysis problem requires the conversion of a sequence of input power images, to a sequence of output images of temperature contours, and this problem is addressed using an *LSTM-based EDGe network* [22].

We describe these networks in the rest of this section.

2.2 U-Nets for Static Analysis

2.2.1 Overview of U-Nets CNNs are successful in extracting 2-D spatial information for image classification and image labeling tasks, which have low-dimensional outputs (class or label). For

PDN and thermal analysis tasks, the required outputs are high-dimensional distributions of IR drop and temperature contours, where the dimensionality corresponds to the number of pixels of the image and the number of pixels is proportional to the size of the chip. This calls for a generator network that can translate the extracted low-dimensional power, PDN, and effective distance to pad features from a CNN-like encoder back into high-dimensional data representing the required output.

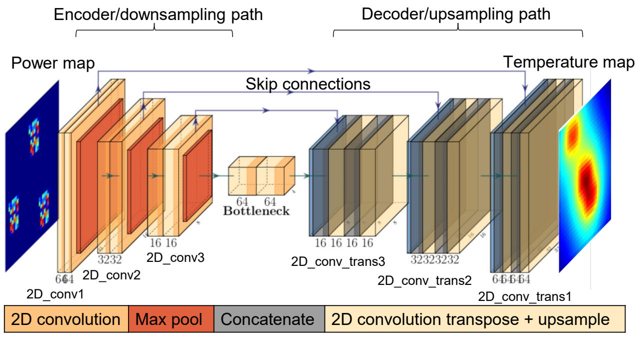

Figure 3: U-Net-based static IREDGe and ThermEDGe.

Fig. 3 shows the structure of the EDGe network used for static PDN and thermal analysis. It consists of two networks:

(a) *Encoder/Downsampling Network* Like a CNN, the network utilizes a sequence of 2-D convolution and max pooling layer pairs that extract key features from the high-dimensional input feature set. The convolution operation performs a weighted sum on a sliding window across the image [23], and the max pooling layer reduces the dimension of the input data by extracting the maximum value from a sliding window across the input image. In Fig. 3, the feature dimension is halved at each stage by each layer pair, and after several such operations, an encoded, low-dimensional, compressed representation of the input data is obtained. For this reason, the encoder is also called the downsampling path: intuitively, downsampling helps understand the *"what"* (e.g., "Does the image contain power or IR hotspots?") in the input image but tends to be imprecise with the *"where"* information (e.g., the precise locations of the hotspots). The latter is recovered by the decoder stages.

(b) *Decoder/Upsampling Network* Intuitively, the generative decoder is responsible for retrieving the *"where"* information that was lost during downsampling, This distinguishes an EDGe network from its CNN counterpart. The decoder is implemented using the transpose convolution [23] and upsampling layers. Upsampling layers are functionally the opposite of a pooling layer, and increase the dimension of the input data matrix by replicating rows and columns.

2.2.2 Use of Skip Connections

Static IR drop and temperature are strongly correlated to the input power – a region with high power on the chip could potentially have an IR or temperature hotspot in its vicinity. U-Nets [14] utilize *skip* connections between the downsampling and upsampling paths, as shown in Fig. 3. These connections take information from one layer and incorporate it using a *concatenation* layer at a deeper stage skipping intermediate layers, and appends it to the embedding along the z-dimension.

Skip connections combine the local power, PDN information, and power pad locations from the downsampling path with the global power information from the upsampling path, allowing the underlying input features to and directly shuttle to the layers closer to the output. This helps recover the fine-grained (*"where"*) details that are lost in the encoder network (as stated before) during upsampling in the decoder for detailed temperature and IR drop contours.

2.2.3 Receptive Fields in the Encoder and Decoder Networks

The characteristic of PDN and thermal analyses problems is that the IR drop and temperature at each location depend on both the local and global power information. During convolution, by sliding averaging windows of an appropriate size across the input power image, the network captures local spatially correlated distributions. For capturing the larger global impact of power on temperature and IR drop, max pooling layers are used after each convolution to appropriately increase the size of the *receptive field* at each stage of the network. The *receptive field* is defined as the region in the input 2-D space that affects a particular pixel, and it determines the impact local, neighboring, and global features have on the analyses.

In a deep network, the value of each pixel feature is affected by all of the other pixels in the receptive field at the previous convolution stage, with the largest contributions coming from pixels near the center of the receptive field. Thus, each feature not only captures its receptive field in the input image, but also gives an exponentially higher weight to the middle of that region [24]. This matches with our applications, where both thermal and IR maps for a pixel are most affected by the features in the same pixel, and partially by features in nearby pixels, with decreasing importance for those that are farther away. The size of the receptive field at each stage in the network is determined by the filter sizes and the number of the convolutional and max pooling layers.

On both the encoder and decoder sides in Fig. 3, we use three stacked convolution layers, each followed by 2×2 max-pooling to extract the features from the power and PDN density images. The number of layers and filter sizes are determined based on the magnitude of the hotspot size encountered during design iterations.

2.3 LSTM-based Model for Transient Analysis

Long short term memory (LSTM) based EDGe networks are a special kind of recurrent neural network (RNN) that are known to be capable of learning long term dependencies in data sequences, i.e., they have a memory component and are capable of learning from past information in the sequence.

For transient thermal analysis, the structure of ThermEDGe is shown in Fig. 4. The core architecture is an EDGe network, similar to the static analysis problem described in Section 2.2, except that the network uses additional LSTM cells to account for the time-varying component. The figure demonstrates the time-unrolled LSTM where input power frames are passed to the network one frame at a time. The LSTM cell accounts for the history of the power maps to generate the output temperature frames. The network is used for sequence-to-sequence translation in transient thermal analysis, where the input is a set of time-varying power maps and the output is a set of time-varying temperature maps (Section 2.1).

Similar to the static ThermEDGe network (Fig. 3), the encoder consists of convolution and max pooling layers to downsample

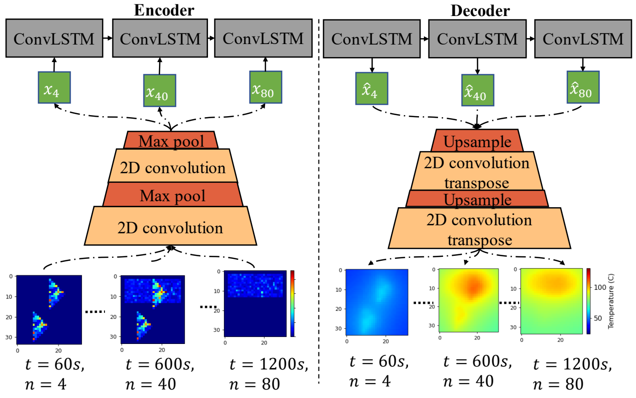

Figure 4: LSTM-based network for transient ThermEDGe.

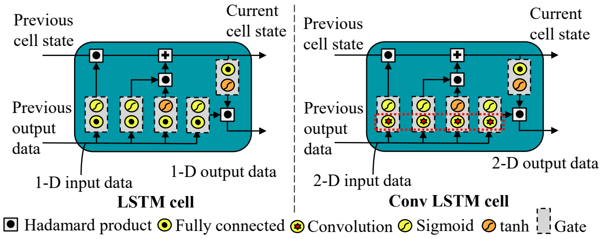

Figure 5: Conventional (left) and ConvLSTM (right) cells.

and extract critical local and global spatial information and the decoder consists of upsampling and transpose convolution layers to upsample the encoded output. However, in addition, transient ThermEDGe has LSTM layers in both the encoder and decoder.

A standard LSTM cell is shown in Fig. 5 (left). While the basic LSTM cell uses fully connected layers within each gate, our application uses a variation of an LSTM cell called a convolutional LSTM (ConvLSTM) [25], shown in Fig. 5 (right). In this cell, the fully connected layers in each gate are replaced by convolution layers that capture spatial information. Thus, the LSTM-based EDGe network obtains a spatiotemporal view that enables accurate inference.

3 EDGe Network Training

We train the models in ThermEDGe and IREDGe to learn the temperature and IR contours from the "golden" commercial tool-generated or ground truth data. We train ThermEdge using the full physics-based thermal simulations from the Ansys-Icepak [26] simulator, incorporating off-chip thermal dynamics from package and system thermal characteristics. IREDGe is trained using static IR drop distribution from a PDN analyzer [27] for various power, PDN density, and power pad distributions.

3.1 Generating Training Data

Static ThermEDGe and IREDGe A challenge we faced to evaluate our experiments is the dearth of public domain benchmarks that fit these applications. The IBM benchmarks [28], are potential candidates for our applications, but they assume constant currents per region and represent an older technology node. Therefore, we generate our dataset which comprises of industry-relevant testcases, where each testcase represents industry-standard workloads for commercial designs implemented in a FinFET technology. The power images are of size 34×32 pixels, with each pixel representing the power/temperature a 250μm×250μm tile on an 8.5mm×8mm chip.[1] Our training is specific to the resolution (250μm×250μm): for another image resolution, the model must be retrained. We reiterate that although the training is performed on chips of fixed size, as we show (Section 4), inference can be performed on a chip of any size as long as the resolution and technology remains the same.

For static ThermEDGe our training data is based on static Ansys-Icepak [26] simulations of these testcases. For IREDGe, we synthesize irregular PDNs of varying densities for each dataset element using *PDN templates*, as defined by OpeNPDN [19]. These templates are a set of PDN building blocks, spanning multiple metal layers in a 14nm commercial FinFET technology, which vary in their metal utilization. For our testcases, we use three templates (high, medium, and low density) and divide the chip into nine regions. As outlined in Section 2.1, we use a checkerboard pattern of power pads that vary in the bump pitch and offsets across the dataset.

The synthesized full-chip PDN, power pad locations, and power distributions are taken as inputs into the IR analyzer [27] to obtain training data for IREDGe. We create a dataset with 5000 datapoints with a combination of 50 different power distributions, 10 different PDN densities, and 10 different patterns of power pad distributions.

Transient ThermEDGe For the transient analysis problem, our training data consists of 150 datapoints with time-varying workloads as features, and the time-varying temperatures from Ansys-Icepak [26] as labels. For each testcase, we generate 45 time-step simulations that range from 0 to 3000s, with irregular time intervals from the thermal simulator. Each simulation to create an element of the training dataset is expensive in terms of the time and memory resources: one simulation of a 3000s time interval with 45 time-steps can take 4 hours with 2 million nodes. Transient ThermEDGe is trained using constant time steps of 15s which enables easy integration with existing LSTM architectures which have an implicit assumption of uniformly distributed time steps, without requiring additional features to account for the time.

3.2 Model Training

For the static analysis problem, ThermEDGe uses a static power map and IREDGe uses a static power map, PDN density map, and effective distance to power pad map as input to predict the corresponding temperature and IR drop contours respectively. For the transient thermal analysis problem, the input is a sequence of 200 power maps and the output is a sequence of 200 temperature contours maps at a 15s time interval. The ML training hyperparameters used for these models are listed in Table 1. The static ThermEDGe and IREDGe models have 132,000 trainable parameters each and the transient ThermEDGe model has 235,000 parameters. It is important to note that the same trained model can be utilized for any chip size as long as the resolution is the same as that of the training data. In addition, the number of parameters in the model is independent of the size of the chip but scales with the size of the hotspot which are generally similar for a given application domain, technology, and packaging choice. A change in hotspot size or resolution demands

[1] Note that although the temperature and power map work at this resolution, the actual simulation consists of millions of nodes; using fewer node (e.g.s, one node per pixel) is grossly insufficient for accuracy.

a change in the number of layers and receptive field sizes of the models to accurately capture temperature and IR drop contours.

Table 1: ThermEDGe and IREDGe ML hyperparameters

ML hyperparameters			Static ThermEDGe	IREDGe	Transient ThermEDGe
Model layer parameters	2D_conv1	filter size	5×5	3×3	5×5
	2D_conv_trans1	# filters	64	64	64
	2D_conv2	filter size	3×3	3×3	3×3
	2D_conv_trans2	# filters	32	32	32
	2D_conv3	filter size	3×3	3×3	–
	2D_conv_trans3	# filters	16	16	–
	Max pool layers	filter size	2×2	2×2	2×2
	ConvLSTM	filter size	–	–	7×7
		# filters	–	–	16
Training parameters	Epochs		500		
	Optimizer		ADAM		
	Loss function		Pixelwise MSE		
	Decay rate		0.98		
	Decap steps		1000		
	Regularizer		L2		
	Regularization rate		1.00E-05		
	Learning rate		1.00E-03		

We split the data in each set for training, validation, and test sets. The training dataset is normalized by subtracting the mean and dividing by the standard deviation and is used to train the network using an ADAM optimizer [29] where the loss function is a pixel-wise mean square error (MSE). The convolutional operation in the encoder and the transpose convolution in the decoder are each followed by ReLU activation to add non-linearity and L2 regularization to prevent over fitting. The model is trained in Tensorflow 2.1 on an NVIDIA GeForce RTX2080Ti GPU. Training run-times are: 30m each for static ThermEDGe and IREDGe, and 6.5h for transient ThermEDGe. We reiterate that this is a one-time cost for a given technology node and package which is amortized over repeated use over many design iterations for multiple chips.

4 Evaluation of TherEDGe and IREDGe

4.1 Experimental Setup and Metric Definitions

ThermEDGe and IREDGe are implemented using Python3.7 within a Tensorflow 2.1 framework. We test the performance of our models on 25 datapoints reserved in the testset (Section 3.2), labeled T1–T25. As mentioned in Section 3.1, due the unavailability of new, public domain benchmarks to evaluate our experiments, we use benchmarks that represent commercial industry design workloads. **Error Metrics** As a measure of goodness of ThermEDGe and IREDGe predictions, we define a discretized regionwise error, $T_{err} = \left|T_{true} - T_{pred}\right|$, where T_{true} is ground truth image, generated by commercial tools, and T_{pred} the predicted image, generated by ThermEDGe. IR_{err} is computed in a similar way. We report the average and maximum values of T_{err} and IR_{err} for each testcase. In addition, the percentage mean and maximum error are listed as a fraction of a temperature corner, i.e., 105°C for thermal analysis and as a fraction of VDD= 0.7V for IR drop analysis.

4.2 Accuracy and Speed

Static ThermEDGe Results A comparison between the commercial tool-generated temperature and the ThermEDGe-generated temperature map for T1–T5 are listed in Table 2. The runtime of static ThermEDGe for each the five testcases which are of size 34×32 is approximately 1.1ms in our environment. Across the five testcases (five rows of the table), ThermEDGe has an average T_{err} of 0.63°C and a maximum T_{err} of 2.93°C.[2] These numbers are a small fraction of the maximum ground truth temperature of these testcases (85 – 150°C). The fast runtimes imply that our method can be used in the inner loop of a thermal optimizer, e.g., to evaluate various chip configurations under the same packaging solution (typically chosen early in the design process). For such applications, this level of error is very acceptable.

Table 2: Results of ThermEDGe across 10 testcases.

Static ThermEDGe			Transient ThermEDGe		
Test	**Avg. T_{err} (C)**	**Max T_{err} (C)**	**Test**	**Avg. T_{err} (C)**	**Max T_{err} (C)**
T1	0.64 (0.61%)	2.76 (2.63%)	T6	0.51 (0.49%)	5.59 (5.32%)
T2	0.63 (0.60%)	2.67 (2.54%)	T7	0.58 (0.55%)	6.17 (5.88%)
T3	0.65 (0.62%)	2.93 (2.79%)	T8	0.57 (0.54%)	5.83 (5.55%)
T4	0.48 (0.46%)	2.22 (2.11%)	T9	0.52 (0.50%)	6.32 (6.02%)
T5	0.75 (0.71%)	2.86 (2.72%)	T10	0.56 (0.53%)	7.14 (6.80%)

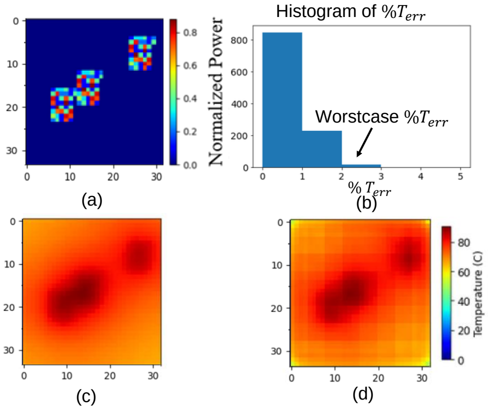

Figure 6: ThermEDGe static temperature estimation on T1: (a) input normalized power distribution, (b) histogram of T_{err}, (c) ground truth, and (d) predicted temperature map.

A graphical view of the predicted map for T1 is depicted in Fig. 6. For a given input power distribution (Fig. 6(a)), we compare the true temperature in Fig. 6(c) against the ThermEDGe-generated temperature contours plots in Fig. 6(d). The discrepancy is visually seen to be small. Numerically, the histogram in Fig. 6(b) shows the distribution of $\%T_{err}$ where the worstcase $\%T_{err}$ is 2.63%.

Transient ThermEDGe Results Trained transient ThermEDGe predicts the output 200-frame temperature sequence at a 15s interval for the input power sequence. We summarize the results in Table 2. The inference runtimes of T6–10 for a sequence 200 frames of temperature contours is approximately 10ms in our setup. Across the five testcases, the prediction has an average T_{err} of 0.52% and a maximum T_{err} of 6.80% as shown. The maximum T_{err} in our testcases occur during transients which do not have long-last effects (e.g., on IC reliability). These errors are reduced to the average T_{err} values at sustained peak temperatures.

Fig. 7 (left) shows a single frame of a video with time-varying power map for T6, where each frame (time-step) is after a 15s time

[2] Achieving this accuracy requires much finer discretization in Icepak.

interval. The corresponding ground truth and predicted temperature contours are in center and right, respectively, of the figure.

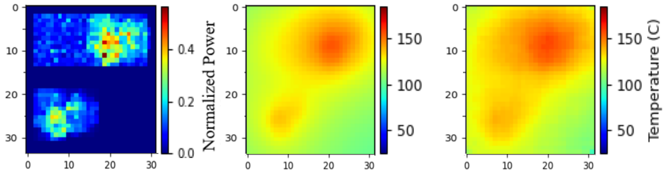

Figure 7: A single frame of a video comparing ThermEDGe prediction (right) against commercial tool (center) temperature contours for T6 power map (left). ***[For an animated version, visit the GitHub repository [17]]***

IREDGe Results We compare IREDGe-generated contours against the contours generated by [27]. Across the five testcases in Table 3, IREDGe has an average IR_{err} of 0.053mV and a worstcase max IR_{err} of 0.34mV which corresponds to 0.008% and 0.048% of VDD respectively. Given that static IR drop constraints are 1–2.5% of VDD, a worstcase error of 0.34mV is acceptable in light of rapid runtimes. We list the results of the testcases in Table 3 where the percentage errors in IR_{err} are listed as fraction of VDD= 0.7V.

Table 3: Results of IREDGe for 10 different testcases. T16-T20 have a chip size that was not in the training set.

Chip size: 34x32			Chip size: 68x32		
	Avg. IR_{err} (mV)	Max IR_{err} (mV)		Avg. IR_{err} (mV)	Max IR_{err} (mV)
T11	0.052 (0.007%)	0.26 (0.03%)	T16	0.035 (0.005%)	0.16 (0.02%)
T12	0.074 (0.011%)	0.34 (0.05%)	T17	0.054 (0.008%)	0.42 (0.06%)
T13	0.036 (0.005%)	0.21 (0.03%)	T18	0.035 (0.005%)	0.35 (0.05%)
T14	0.053 (0.008%)	0.24 (0.03%)	T19	0.068 (0.010%)	0.22 (0.03%)
T15	0.051 (0.007%)	0.23 (0.03%)	T20	0.061 (0.009%)	0.38 (0.05%)

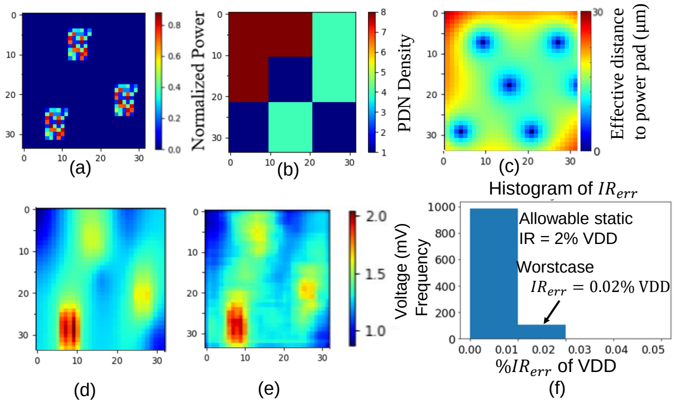

Figure 8: IREDGe data for T11: input (a) power map, (b) PDN density map, (c) effective distance map, output (d) ground truth, (e) predicted IR drop map, and (f) histogram of IR_{err}.

A detailed view of T11 is shown in Fig. 8. It compares the IREDGe-generated IR drop contour plots against contour plot generated by [27]. The input power maps, PDN density maps, and effective distance to power pad maps are shown in Fig. 8(a), (b), and (c) respectively. Fig. 8(d) and (e) shows the comparison between ground truth and predicted value for the corresponding inputs. It is evident that the plots are similar; numerically, the histogram in Fig. 8(f) shows the $\%IR_{err}$ where the worst $\%IR_{err}$ is less than 0.02% of VDD.

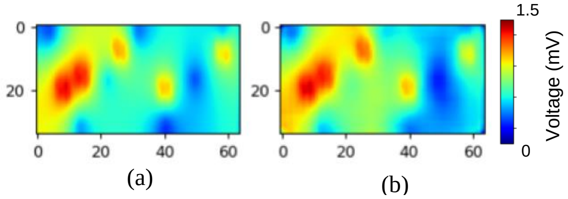

Figure 9: Comparison between actual (left) and IREDGe-predicted (right) IR drop contours for images of size 68×32 using a model that was trained on images of size 34×32.

Size-independence Since the EDGe models only comprise the trained weights of the convolutional kernels, the same model can be reused to predict IR drop or temperature contours of a chip of a different size. We demonstrate this using IREDGe on chips of a different size (T16 – T20), using an input power distribution of size 68 × 32. Fig. 9 compares the actual IR drop of T16 (left) and the IREDGe-predicted (right) using a model which was trained on 34 × 32 power maps. The results on T16 – T20 are in Table 3.

Inference Runtime Analysis Table 4 compares the inference runtimes of our ML-based EDGe network approach against the golden solvers. The runtimes are reported on a NVIDIA GeForce RTX 2080Ti GPU. With the millisecond inference times, and the transferable nature of our trained models, the one-time cost of training the EDGe networks is easily amortized over multiple uses within a design cycle, and over multiple designs.

Table 4: Runtimes of EDGe networks and golden analysis

Analysis type	**# Nodes**	**Design Area**	**Icepak/ PDNSim runtimes**	**ThermEDGe/ IREDGe inference runtimes**
Static thermal	2.0 million	68 mm^2	30 mins	1.1 ms
Transient thermal	2.0 million	68 mm^2	210 mins	10 ms
Static IR drop	5.2 million	0.16 mm^2	5 mins	1.1 ms

4.3 IREDGe Compared with PowerNet

We compare IREDGe against our implementation of PowerNet, based on its description in [9]. The layout is divided into tiles, and the CNN features are the 2-D power distributions (toggle rate-scaled switching and internal power, total power, and leakage power) within each tile and in a fixed window of surrounding tiles. The trained CNN is used to predict the IR drop on a tile-by-tile basis by sliding a window across all tiles on the chip. The work uses a tile size of 5μm×5μm and takes into consideration a 31×31 tiled neighborhood (window) power information as features. For a fair comparison, we train IREDGe under a fixed PDN density and fixed power pad locations that is used to train PowerNet. Qualitatively, IREDGe is superior on three aspects:

(1) *Tile and Window Size Selection:* It is stated in [9] that when the size of the tile is increased from 1μm×1μm to 5μm×5μm and the size of the resulting window is increased to represent 31×31

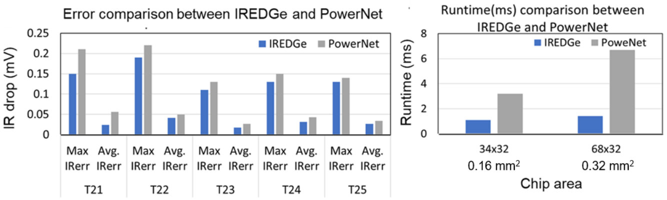

Figure 10: IREDGe versus PowerNet: IR drop error (left) and runtimes (right). IREDGe is 2.9× faster at iso-error across T21–T25 (0.16mm^2 area).

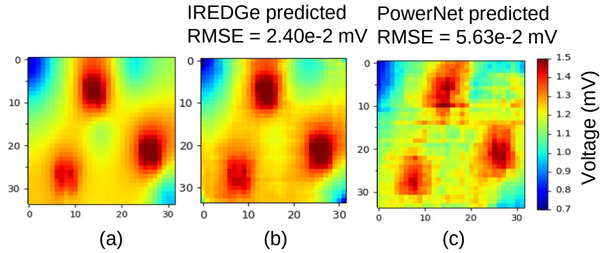

Figure 11: IR drop comparisons on T21: (a) ground truth, (b) IREDGe, and (c) our implementation of PowerNet.

window of 25μm^2 tiles instead of 1μm^2 tiles, the accuracy of the PowerNet model improves. In general, this is the expected behavior with an IR analysis problem where the accuracy increases as more global information is available, until a certain radius after which the principle of locality holds [12]. IREDGe bypasses this tile-size selection problem entirely by providing the entire power map as input to IREDGe and allowing the network to learn the window size that is needed for accurate IR estimation.

(2) *Runtimes and accuracy:* We compare IREDGe against our implementation of PowerNet on T21–25. These testcases have the same power distributions in T11–15 except that all the five testscases have identical uniform PDNs, and identical power pad distributions, as required by PowerNet; IREDGe does not require this. Unlike PowerNet, which trains and infers IR drop on a sliding tile-by-tile basis, IREDGe has faster training and inference. IREDGe requires a *single* inference, irrespective of the size of the chip while PowerNet performs an inference for every tile in the chip. For this setup and data, it takes 75 minutes to train PowerNet, as against 30 minutes for IREDGe. Fig. 10 shows the comparison of inference times between PowerNet and IREDGe at similar error levels. At the 25μm^2 tile size, both IREDGe and PowerNet have similar accuracies for T21–25 as shown in Fig. 10(a). At this error level, for 0.16 mm^2 designs, IREDGe is 2.9× faster than PowerNet (Fig. 10(a)).

(3) *Piexelated IR drop maps:* Since PowerNet uses a CNN to predict IR drop on a tile-by-tile basis, where each tile is 5μm × 5μm, the resulting IR drop image is pixelated, and the predicted region value does not correlate well with the neighboring regions. This is highlighted in Fig. 11 which compares IR drop contours from a golden solver (Fig. 11(a)), IREDGe (Fig. 11(b)), and our implementation of PowerNet (Fig. 11(c)) for T21.

5 Conclusion

This paper addresses the compute-intensive tasks of thermal and IR analysis by proposing the use EDGe networks as apt ML-based solutions. We successfully evaluate EDGe networks for these applications by developing two ML-based tools (i) ThermEDGe and (ii) IREDGe for rapid on-chip (static and dynamic) thermal and (static) IR analysis respectively. In principle, our methodology is applicable to dynamic IR as well, but could not be exercised here due to the unavailability of public-domain benchmarks. In forthcoming work, we expand this method to make it more scalable and demonstrate it on industry circuits for dynamic IR analysis [30].

References

[1] Y. Zhan, *et al.*, "Thermally-aware design," *Found. Trends Electron. Des. Autom.*, vol. 2, no. 3, pp. 255–370, 2008.
[2] Y. Zhong and M. D. F. Wong, "Fast algorithms for IR drop analysis in large power grid," in *Proc. ICCAD*, 2005.
[3] J. N. Kozhaya, *et al.*, "A multigrid-like technique for power grid analysis," *IEEE T. Comput. Aid. D.*, vol. 21, Oct. 2002.
[4] K. Zhang, *et al.*, "Machine learning-based temperature prediction for runtime thermal management across system components," *IEEE Trans Parallel Distrib. Syst.*, vol. 29, Feb. 2018.
[5] D. Juan, *et al.*, "A learning-based autoregressive model for fast transient thermal analysis of chip-multiprocessors," in *Proc. ASP-DAC*, 2012.
[6] S. Sadiqbatcha, *et al.*, "Hot spot identification and system parameterized thermal modeling for multi-core processors through infrared thermal imaging," in *Proc. DATE*, 2019.
[7] S.-Y. Lin, *et al.*, "IR drop prediction of ECO-revised circuits using machine learning," in *Proc. VTS*, 2018.
[8] C. Ho and A. B. Kahng, "IncPIRD: Fast learning-based prediction of incremental IR drop," in *Proc. ICCAD*, 2019.
[9] Z. Xie, *et al.*, "PowerNet: Transferable dynamic IR drop estimation via maximum convolutional neural network," in *Proc. ASP-DAC*, 2020.
[10] W. Jin, *et al.*, "Full-chip thermal map estimation for commercial multi-core CPUs with generative adversarial learning," in *Proc. ICCAD*, pp. 1–9, 2020.
[11] J. Wen, *et al.*, "DNN-based fast static on-chip thermal solver," in *Semicond. Therm. Meas., Model. Manage. Symp.*, 2020.
[12] E. Chiprout, "Fast flip-chip power grid analysis via locality and grid shells," in *Proc. ICCAD*, 2004.
[13] E. Shelhamer, *et al.*, "Fully convolutional networks for semantic segmentation," *IEEE T. Pattern Anal. Mach. Intell.*, vol. 39, Apr. 2017.
[14] O. Ronneberger, *et al.*, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assisted Intervention*, 2015.
[15] X. Mao, *et al.*, "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," in *Proc. NeurIPS*, 2016.
[16] V. Badrinarayanan, *et al.*, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE T. Pattern Anal. Mach. Intell.*, vol. 39, Dec. 2017.
[17] V. A. Chhabria, *et al.*, "ThermEDGe-and-IREDGe." github.com/VidyaChhabria/ThermEDGe-and-IREDGe.
[18] J. Singh and S. S. Sapatnekar, "Partition-based algorithm for power grid design using locality," *IEEE T. Comput. Aid D.*, vol. 25, Apr. 2006.
[19] V. A. Chhabria, *et al.*, "Template-based PDN synthesis in floorplan and placement using classifier and CNN techniques," in *Proc. ASP-DAC*, 2020.
[20] B. W. Amick, *et al.*, "Macro-modeling concepts for the chip electrical interface," in *Proc. DAC*, 2002.
[21] F. Yazdani, "Foundations of heterogeneous integration: An industry-based, 2.5D/3D pathfinding and co-design approach," (Boston, MA), Springer, 2018.
[22] I. Sutskever, *et al.*, "Sequence to sequence learning with neural networks," in *Proc. NeurIPS*, 2014.
[23] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv:1603.07285v2*, Mar. 2016.
[24] W. Luo, *et al.*, "Understanding the effective receptive field in deep convolutional neural networks," in *Proc. NeurIPS*, 2016.
[25] X. Shi, *et al.*, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Proc. NeurIPS*, 2015.
[26] "Ansys-Icepak." www.ansys.com/products/electronics/ansys-icepak.
[27] V. A. Chhabria and S. S. Sapatnekar, "PDNSim." https://github.com/The-OpenROAD-Project/OpenROAD/tree/master/src/PDNSim.
[28] S. R. Nassif, "Power grid analysis benchmarks," in *Proc. ASP-DAC*, 2008.
[29] D. Kingma and J. Ba, "ADAM: A method for stochastic optimization," in *Proc. ICLR*, 2014.
[30] V. A. Chhabria, *et al.*, "MAVIREC: ML-aided vectored IR-drop estimation and classification," in *Proc. DATE*, 2021.