On Robustness and Generalization of ML-Based Congestion Predictors to Valid and Imperceptible Perturbations

Chester Holtz*
chholtz@eng.ucsd.edu
University of California San Diego

Yucheng Wang
yuw132@ucsd.edu
University of California San Diego

Chung-Kuan Cheng
ckcheng@eng.ucsd.edu
University of California San Diego

Bill Lin
billlin@eng.ucsd.edu
University of California San Diego

ABSTRACT

There is substantial interest in the use of machine learning (ML)-based techniques throughout the electronic computer-aided design (CAD) flow, particularly methods based on deep learning. However, while deep learning methods have achieved state-of-the-art performance in several applications (e.g. image classification), recent work has demonstrated that neural networks are generally vulnerable to small, carefully chosen perturbations of their input (e.g. a single pixel change in an image). In this work, we investigate robustness in the context of ML-based EDA tools—particularly for congestion prediction. As far as we are aware, we are the first to explore this concept in the context of ML-based EDA.

We first describe a novel notion of imperceptibility designed specifically for VLSI layout problems defined on netlists and cell placements. Our definition of imperceptibility is characterized by a guarantee that a perturbation to a layout will not alter its global routing. We then demonstrate that state-of-the-art CNN and GNN-based congestion models exhibit brittleness to imperceptible perturbations. Namely, we show that when a small number of cells (e.g. 1%—5% of cells) have their positions shifted *such that a measure of global congestion is guaranteed to remain unaffected* (e.g. 1% of the design adversarially shifted by 0.001% of the layout space results in a predicted decrease in congestion of up to 90%, while no change in congestion is implied by the perturbation). In other words, the quality of a predictor can be made arbitrarily poor (i.e. can be made to predict that a design is "congestion-free") for an arbitrary input layout. Next, we describe a simple technique to train predictors that improves robustness to these perturbations. Our work indicates that CAD engineers should be cautious when integrating neural network-based mechanisms in EDA flows to ensure robust and high-quality results.

1 INTRODUCTION

Electronic design automation (EDA) flows involve significant optimization and verification challenges that continue to scale as the complexity of designs increases. There is substantial interest in using machine learning techniques for solving electronic computer-aided design (CAD) problems ranging from logic synthesis to physical design and design for manufacturability (DFM) [15]. Prior work has demonstrated that deep learning-enhanced design flows are faster and more scalable, particularly when integrated to augment the time-consuming stages of layout [5, 13, 20], design space exploration [11], logic optimization [21] and lithographic analysis [22].

However, although neural networks have been extremely successful in the aforementioned EDA tasks, recent work [10, 18] has demonstrated that image classifiers can be fooled by small, carefully chosen perturbations of their input. Notably, Su et al. [17] demonstrated that neural network classifiers which can correctly classify "clean" images may be vulnerable to *targeted attacks*, e.g., misclassify those same images when only a single pixel is changed.

The question that we aim to explore in this work is the following:

> ***To what degree are neural network-based congestion predictors vulnerable to small, but valid, changes in layout input?***

As the application of machine learning to production EDA tasks becomes more widespread, understanding and addressing this question will become increasingly critical. In this work, we provide evidence that supports an affirmative answer:

> ***Congestion predictors erroneously predict large changes to routing congestion with respect to changes to the layout that do not change the global routing.***

Specifically, we investigate a novel notion of validity and design two efficient methods for finding perturbations that demonstrate brittleness of recently proposed congestion predictors. Furthermore, we describe one potential approach to address the highlighted issues and demonstrate that modifying the training procedure to promote robustness is one promising direction to address brittleness to imperceptible changes. Although we focus on congestion prediction, our work generalizes to arbitrary predictive models integrated in EDA pipelines. More generally, our work motivates the need for careful evaluation of the generalization of ML-based EDA tools—in excess of typical performance metrics reported on a train-test split.

1.1 Contributions

The primary contribution of this work is to demonstrate that modern deep learning-based EDA tools—specifically congestion predictors—are vulnerable to valid perturbations to their inputs, i.e. may exhibit poor generalization to perturbations of the cell layout.

Inspired by the perspective of *adversarial perturbations*, given an input design layout, we characterize *small* perturbations as (1.) perturbations that result in the adjustment of relatively few cell positions and (2.) perturbations that maintain the global congestion structure (Sec. 3). We describe a numerical algorithm to efficiently

*Corresponding author

search the *feasible adversarial neighborhood* of an input to find small perturbations that maintain validity of the input design while drastically reducing the efficacy of ML-based EDA tool predictions (Sec. 3). We describe two variants of the proposed method: while both rely on knowledge of the predictor weights—known as a white-box perturbation model—one method requires knowledge of the underlying congestion structure, while the second method does not necessitate such information. We then demonstrate (1.) that congestion predictors are vulnerable to both models and (2.) that *adversarial training* significantly improves robustness with only a modest performance trade-off. We emphasize that while we frame our discussion in the context of robustness to perturbations, our findings motivate a broader need to study implications of poor generalization when integrating ML-based tools into design flows.

In summary, our contributions include the following:

(1) A novel formulation of the *feasible neighborhood* of an input design—i.e. given a layout, what small perturbations maintain the relevant measures of congestion?
(2) Efficient supervised and unsupervised algorithms for computationally searching the neighborhood of a layout.
(3) Exploration of adversarial training as a way to induce robustness and improve generalization.
(4) Under a previously defined characterization of predictive quality for congestion tasks [1], we show that the benchmark layouts we evaluated can be perturbed such that the congestion predictions on the perturbed layouts are poor.

2 PRELIMINARIES AND RELATED WORK

In this section, we provide an overview of ML-based EDA methods and adversarially robust prediction. Let $x, y \in \mathbb{R}^n$ be vectors corresponding to the coordinates of n components such that the i-th component has coordinates encoded in the i-th row of $X := [x : y]$; $[x : y]_i$. We aim to find perturbations to the layout so that the resulting layout satisfies certain constraints (i.e. remains in the neighborhood of the original layout with respect to global routing).

Number of components	$n \in \mathbb{R}_+$
Placement coordinates	$x, y \in \mathbb{R}^n$, $X = [x : y] \in \mathbb{R}^{n\times 2}$
Placement perturbation	$\delta_x, \delta_y \in \mathbb{R}^n$, $\Delta = [\delta_x : \delta_y] \in \mathbb{R}^{n\times 2}$
Neural network parameters	θ
Early global routing bins	$W \times H$
Feature map	$M \in \mathbb{R}^{W\times H}$
Predicted congestion map	$f_\theta(M)$

Figure 1: Notation

2.1 Global routing

The VLSI routing problem is usually solved in two steps: (1.) global routing and (2.) detailed routing. The principle aim of the global routing step is to generate a routing solution on a discretization of the layout space, represented as a grid graph and provide a preferred routing region (i.e. a route guide) for the detailed router.

A typical multi-commodity flow formulation of global routing partitions the routing space into regular rectangles (G-Cells) and generates a grid graph $G = (V, E)$ in which each vertex $v \in V$ represents a G-Cell and each edge $e \in E$ represents the connection between adjacent G-Cells. The capacity of an edge represents the maximum number of wires that can go through the edge and the variable assignment of the edge corresponds the number of wires that are currently using the edge, while the overflow is denoted by the number of wires that exceeds the capacity. For each net, the routing problem is to find a path that connects all the pins of a net in the given grid graph while avoiding overflow on the edges. In other words, the global router maximizes a measure of routability with respect to the detailed router while satisfying certain constraints to manage design rule violations (DRVs), pin accessibility, and irregular module geometries. An important concept is the construction of the graph G. The graph, global routing solution, and associated congestion metrics remain consistent as long as individual cells remain within their G-Cells, regardless of their precise positions within each G-Cell.

2.2 RUDY

Rectangular Uniform wire Density (RUDY) [16] is a method to estimates the wirelength density by uniformly spreading the wire volume of nets into its bounding box. It is very commonly used as a feature to indicate the relative congestion of a region. The RUDY map of a net e represents the average wirelength per unit area in the bounding box of the net: $\mu^{(e)}(\frac{1}{x_{\max}-x_{\min}} + \frac{1}{y_{\max}-y_{\min}})$ where the net-map $\mu^{(e)} \in \mathbb{R}^{W\times H}$ is

$$\mu_{xy}^{(e)} = \begin{cases} 1 & x_{\min} \le x \le x_{\max} \text{ and } y_{\min} \le y \le y_{\max} \\ 0 & \text{otherwise} \end{cases}$$

$x_{\min}$, and $x_{\max}$ correspond to the maximum and minimum x coordinates of the associated net, and $y_{\min}$ and $y_{\max}$ correspond to the maximum and minimum y-coordinate of the associated net. The RUDY score assigned to a location (x, y) is computed by aggregating RUDY scores over all nets $e \in E$.

2.3 Machine learning and EDA

As previously mentioned, ML-based prediction has been explored for various early-stage tasks in EDA flows including routability, DRC, and IR drop prediction [5, 13, 20]. For the purposes of this work, we focus on the congestion prediction framework proposed by Liu et al. [13]. Notably, Liu et al. [13] use the Innovus global router to obtain ground truth congestion hotspots, while an $W\times H\times 3$ feature map M, comprised of RUDY scores [16], an associated pin-density variant PinRudy, and a macro placement map MacroRegion is derived from the associated cell placement. A neural network is used to learn a mapping from feature maps to congestion hotspots. The authors of [13] also derive the gradients of the unsupervised congestion penalty $\frac{1}{HW}||f_\theta(M)||_F^2$ with respect to cell locations.

2.4 Machine learning and robustness

Consider the network $f : \mathbb{R}^d \to \mathbb{R}^k$, where the input is d-dimensional and the output is a k-dimensional vector of likelihoods. For example, the input could be a d-dimensional image and the j-th entry of the output could correspond to the likelihood the image belongs to the j-th class. The associated prediction is then $c(x;\theta) = \arg\max_{j\in[k]} f_j(x;\theta)$.

Recently, machine learning practitioners have not just been concerned that the prediction be correct, but also want robustness to random or adversarial noise, i.e. small perturbations to the input which may change the prediction to an incorrect class. We define the notion of ϵ-robustness below:

Definition 2.1 (ϵ-robust). f parameterized by θ is called ϵ-robust with respect to norm p at x if the prediction is consistent for a small ball of radius ϵ around x:

$$c(x+\delta;\theta) = c(x;\theta), \quad \forall \delta : ||\delta||_p \leq \epsilon. \tag{1}$$

The minimal ℓ_p-norm perturbation δ_p^* required to switch an sample's label is given by the solution to the following problem:

$$\delta_p^* = \arg\min ||\delta||_p \quad \text{s.t.} \quad c(x;\theta) \neq c(x+\delta;\theta).$$

A significant amount of existing work relies on a first-order approximations and Hölder's inequality to recover δ^*.

Projected Gradient Descent (PGD) is a first-order method that can be used to find an approximation of δ_p^*. PGD-type algorithms consist of a descent step followed by a projection onto the feasible set S. Given the current iterate $x^{(i)}$, the next iterate $x^{(i+1)}$ is computed via a transformation s applied to the gradient of the loss function L. For example, if labels are available to the perturbation algorithm, L could be the original loss used to train the network f. Alternatively, L can be substituted for an unsupervised metric.

$$\begin{aligned} u^{(i+1)} &= x^{(i)} + \eta^{(i)} \cdot s(\nabla L(x^{(i)})) \\ x^{(i+1)} &= P_S(u^{(i+1)}) \end{aligned} \tag{2}$$

$\eta^{(i)} > 0$ the step size at iteration i, $s : \mathbb{R}^d \to \mathbb{R}^d$ determines the descent direction as a function of the gradient of the loss L at $x^{(i)}$ and $P_S : \mathbb{R}^d \to S$ is the projection on S. For example, an ℓ_1-perturbation model of radius ϵ, we denote by $B_\infty(x,\epsilon) = \{z \in \mathbb{R}^d | \ ||z-x||_\infty \leq \epsilon\}$. A crucial choice is that of the descent direction $s(\nabla L(x_i))$, a mapping s applied to a gradient. For example, the steepest descent direction [2]:

$$\delta_p^* = \arg\max_{\delta \in \mathbb{R}^d} \langle w, \delta \rangle \quad \text{s.t.} \quad ||\delta||_p \leq \epsilon \tag{3}$$

with $w = \nabla f(x_i) \in \mathbb{R}^d$, the maximizer of a linear function over a given ℓ_p ball constraint. Thus one gets $\delta_\infty^* = \epsilon \text{sign}(w)$ and $\delta_2^* = \epsilon w / ||w||_2$ for $p = \infty$ and $p = 2$ respectively, which define s.

3 A NOVEL NOTION OF IMPERCEPTIBILITY FOR CONGESTION PREDICTION

In this section, we describe a method to compute layout perturbations that guarantee consistency of the global routing. We present a general illustration of our method in Fig. 2. In other words, given a trained model, we seek to adjust a given layout (the coordinates of a subset of the cells in a design) such that the adjusted layout remains valid, but spoils the congestion predictions of the model.

In particular, we define the feasible set of perturbations that we utilize in the context of congestion predictions and an algorithm to perform a projection onto this set. *Specifically, we discuss perturbations of coordinate-based representations of chip layouts.* Inspired by earlier work on adversarial attacks designed for image classifiers [7–9], we impose a natural set of constraints on the perturbation to ensure that the global routing does not change. Individually, these constraints prohibit cells from moving outside of their G-Cell tiles.

Let $X := [x : y]^\top \in \mathbb{R}^{n \times 2}$ be the coordinate assignments to each cell and $\delta := [\delta_x : \delta_y]^\top \in \mathbb{R}^{n \times 2}$ be a perturbation. For simplicity, and without loss of generality, let us consider the 1st column of X and δ; x, δ_x (i.e. the x-coordinate of each cell and associated perturbations such that the perturbed cell x-coordinates of the layout is given by $x + \delta_x$).

We say that δ_x is an *imperceptible* perturbation in the context of physical design if the following conditions hold:

(1) Global imperceptibility: the maximum number of cells that can be moved is bounded by ϵ_0: $||\delta||_0 \leq \epsilon_0$ (e.g. 1% of the total number of cells.
(2) Local imperceptibility: a cell can only be moved within it's associated G-Cell: $l_i \leq (x+\delta_x)_i \leq u_i$

Given $x \in [0,1]^d$, we define the feasible set—the intersection of the ℓ_0-ball of radius ϵ_0, the upper and lower-bound—*box*—constraints on each entry of x, and the set $[0,1]^d$:

$$\begin{aligned} S(x) = \Big\{ z \in \mathbb{R}^d | \ \sum_{i=1}^d \mathbf{1}_{|z_i - x_i| > 0} \leq \epsilon_0, \\ l_i \leq z_i - x_i \leq u_i, \ \ 0 \leq z_i \leq 1 \Big\} \end{aligned} \tag{4}$$

First, note that intersection of the two constraints $0 \leq z_i \leq 1$ and $l_i \leq z_i - x_i \leq u_i$ may be written:

$$\max\{0, l_i + x_i\} \leq z_i \leq \min\{1, u_i + x_i\}$$

So, the feasible set may be simplified and re-written

$$\begin{aligned} S(x) = \Big\{ z \in \mathbb{R}^d | \sum_{i=1}^d \mathbf{1}_{|z_i - x_i| > 0} \leq \epsilon_0, \\ \max\{0, l_i + x_i\} \leq z_i \leq \min\{1, u_i + x_i\} \Big\} \end{aligned} \tag{5}$$

The Euclidean projection onto $S(x)$; P_S is then defined to be

$$\begin{aligned} \min_z ||y - z||_2^2 \quad \text{s.t.} \ \sum_{i=1}^d \mathbf{1}_{|z_i - x_i| > 0} \leq \epsilon_0, \\ \max\{0, l_i + x_i\} \leq z_i \leq \min\{1, u_i + x_i\} \end{aligned} \tag{6}$$

Ignoring the combinatorial constraint, the solution is given by $z_i^* = \max\{l_i + x_i, \min\{y_i, u_i + x_i\}\}$. We re-integrate the ℓ_0 constraint and resolve the projection by sorting according to the *gain* ϕ:

$$\phi_i = (y_i - x_i)^2 - (y_i - z_i^*)^2, \quad z_{\pi_i} = \begin{cases} z_{\pi_i}^* & i = 1, \ldots, k \\ x_{\pi_i} & \text{otherwise} \end{cases}$$

Thus, the final solution will have k entries—those with the highest gain—and that differ by no more than l_i or u_i.

Importantly, the solution to this problem can be computed efficiently; requiring a single backward pass through the trained model to compute the perturbed layout y and computation of the projection P_S in linear time—note that when ignoring the combinatorial constraint, Prob. 6 is a separable problem (can be computed in parallel) over the cells. Finding the subset of cells to shift (satisfying the ℓ_0 constraint) involves a linear-time scan over gains.

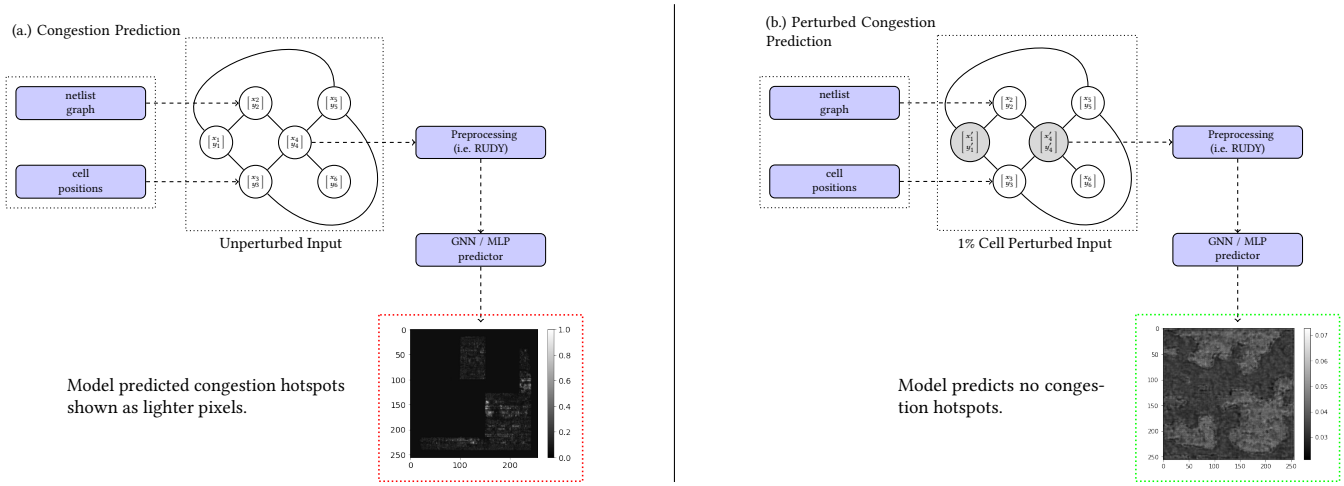


Figure 2: General illustration of effect of imperceptible perturbations on EDA predictions (a.): Vanilla prediction framework. The netlist-graph and cell attributes (i.e. positions) are used to make predictions (e.g. DRC locations or congestion hotspots) via the neural predictor. (b.): The attributes of a subset of nodes are perturbed: $x' = x + \delta_x$. The predictor is vulnerable (i.e. can be made to predict that a design is congestion-free)—even when both δ_x and the number of perturbed nodes are small.

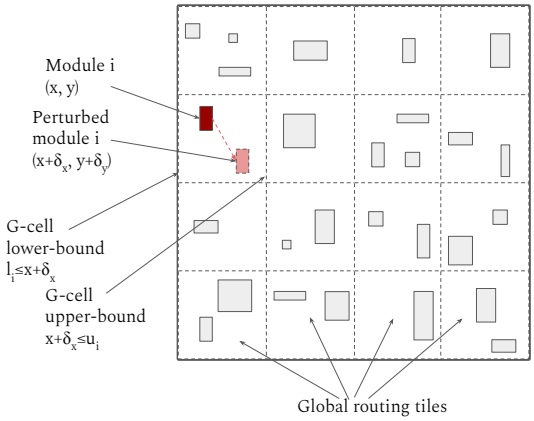


Figure 3: Local constraints for each movable cell (highlighted in red) ensures cells do not move G-Cells.

Table 1: Vulnerability of ML-based EDA predictors to imperceptible perturbations. "*" denotes a perturbation that induces congestion-free predictions. "†" denotes a perturbation that induces mispredictions of hotspots.

	$\frac{1}{HW}\|\|M\|\|_F^2$	NRMS	SSIM	$\frac{1}{HW}\|\|M\|\|_F^2$	NRMS	SSIM
FCN Model	Vanilla training			Robust training		
Vanilla / none	0.010	0.0393	0.8044	0.010	0.0393	0.7970
Random noise	0.012	0.0420	0.7970	0.012	0.0420	0.7123
* 1% cells perturbed	0.0012	0.0791	0.6255	0.011	0.0561	0.6831
* 5% cells perturbed	0.0011	0.0945	0.5152	0.012	0.0533	0.6181
†1% cells perturbed	0.011	0.1055	0.4534	0.011	0.0431	0.7011
†5% cells perturbed	0.011	0.1467	0.4334	0.011	0.0440	0.7193
GNN Model						
Vanilla / none	0.011	0.0348	0.8130	0.011	0.0393	0.7970
Random noise	0.013	0.0384	0.7974	0.011	0.0417	0.6933
* 1% cells perturbed	0.0013	0.0743	0.6129	0.0098	0.0403	0.7643
* 5% cells perturbed	0.0012	0.0892	0.5032	0.0097	0.0407	0.7392
†1% cells perturbed	0.011	0.1744	0.4461	0.011	0.0411	0.694
†5% cells perturbed	0.013	0.1835	0.4219	0.013	0.0417	0.695

4 EXPERIMENTS

In this section we describe a set of comprehensive experiments on testcases from the CircuitNet suite [4]. Summary statistics of the testcases are presented in Table 2. Our numerical experiments are aimed at establishing the efficacy of our method with respect to spoiling congestion predictions made by two trained architectures.

4.1 Experimental setup

We evaluate the impact of our small perturbations on the robustness of ML-based congestion predictors. Moreover, we give illustrative examples of such sparse and imperceivable perturbations. We utilize the CircuitNet benchmarks [4] to validate our method. CircuitNet is an open-source dataset consisting of more than 10K samples from versatile runs of commercial design tools based on open-source RISC-V designs with various features for multiple ML for EDA applications. We summarize the designs and generation procedure used to compose the CircuitNet dataset in Table 2.

Perturbations are generated using a momentum-based PGD algorithm with restarts. We adapt the standard PGD iterations outlined in Eq. 2 in two ways: (1.) we adjust the gradient-based update rule to incorporate a momentum term:

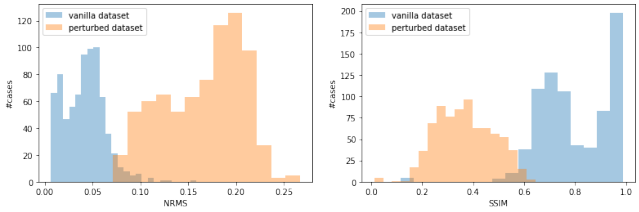
$$\begin{aligned} u^{(i+1)} &= P_S(x^{(i)} + \eta^{(i)} \cdot s(\nabla L(x^{(i)}))) \\ x^{(i)} &= P_S(x^{(i)} + \alpha(u^{(i+1)} - x^{(i)}) + (1-\alpha)(x^{(i)} - x^{(i-1)})) \end{aligned} \tag{7}$$

where $\alpha \in [0, 1]$ regulates the influence of the previous update on the current one and P_S is described in Sec. 3. (2.) we introduce "restarts"—i.e. we apply PGD to several random initializations and select the best solution.

Table 2: From [4]. Statistics of designs and variations.

Design	Netlist Statistics			Synthesis Variations	
	#Cells	#Nets	Cell Area (μm^2)	#Macros	Frequency (MHz)
RISCY-a	44836	80287	65739	3/4/5	50/200/500
RISCY-FPU-a	61677	106429	75985		
zero-riscy-a	35017	67472	58631		
RISCY-b	30207	58452	69779	13/14/15	
RISCY-FPU-b	47130	84676	80030		
zero-riscy-b	20350	45599	62648		

Physical Design Variations			
Utilizations (%)	#Macro Placement	#Power Mesh Setting	Filler Insertion
70/75/80/85/90	3	8	After Placement /After Routing



(a) (b)

Figure 4: Performance metrics associated with a vanilla network evaluated on unperturbed and perturbed layouts. (a) Distribution shift in NRMS. (b) Distribution shift in SSIM. Liu et al. [13] characterize a good predictor as achieving $NRMS < 0.2$ and $SSIM > 0.8$. Using our method, we are able create valid inputs such that approximately 100% of samples have $SSIM < 0.8$ and 60% of samples have $NRMS > 0.15$. 100% of samples satisfy one of the two conditions.

4.1.1 Algorithm parameters. We adopt the same 70/30 train-test split described in the CircuitNet paper. We perturb all samples in the test. In Table 1 we report several metrics for each method including our congestion score, NRMS, and SSIM. We set $\alpha = 0.75$ and fix η to be $2 \cdot w$, where w is the width of each G-Cell. Each perturbed example is generated by running PGD for 100 iterations with 5 restarts, and η is linearly decayed to $1/10 \cdot w$.

4.2 On the robustness of congestion predictors

In Table 1, we evaluate our method for producing valid and imperceptible perturbations using the fully convolutional architecture proposed in Liu et al. [13] and a single-layer graph convolutional network (GNN Model) proposed in [12]. Two variants of our algorithm are implemented. For rows denoted by a *, the perturbation ascent direction is computed with respect to the congestion score $\frac{1}{HW}||f_\theta(M)||_F^2$. For rows denoted by a †, the perturbation ascent direction is computed with respect to the *supervised prediction loss*.

We demonstrate that neural network predictors fail to accurately predict congestion of layouts produced by our method. Furthermore, when a larger budget of cells are allowed to be shifted, performance

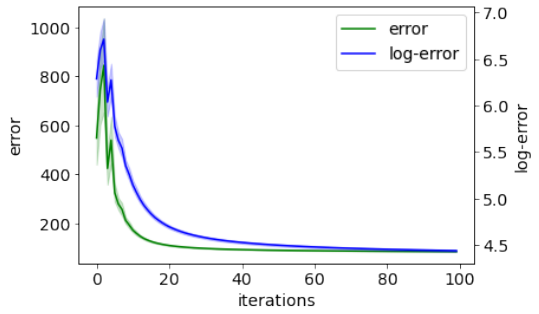


Figure 5: Mean relative unsupervised error of PGD over iterations. Shaded region denotes 1 standard deviation. Note the log-scale in blue implies convergence in error.

is further degraded. In particular, we first observe that vanilla models are relatively robust to a random perturbation model. Namely, we uniformly at random select 1% of cells and maximally perturb their associated positions such that they remain within their associated G-Cell. When the layout is altered in this way, we see that neural predictors generally maintain their performance with only a minor degradation in NRMS and SSIM observed.

Next, we demonstrate that perturbations may be carefully chosen such that the associated predictions correspond to congestion-free predictions, or even adversarial predictions- i.e. the model predicts congestion in regions which are congestion-free and predicts congestion-free regions in areas which are highly congested (e.g. in regions with macros). We provide examples to demonstrate these instances in Figure 7. Distributions of SSIM and NRMS scores are provided in Figure 4. Notably, the predictor violates the conditions necessary for good performance (NRMS < 0.2, SSIM > 0.8) [1, 4].

When a budget of 1% of cells is prescribed, a degradation in SSIM of 43.64% and a degradation in NRMS of 168.45% are observed for the FCNN-based predictor. Likewise, when the budget is increased to 5% of cells, degradations in SSIM and NRMS amount to 46.12% and 273.28% are observed respectively. As expected, the GNN-based model is also vulnerable to the aforementioned issues. Interestingly, while the GNN-based method outperforms the FCN-based method on unperturbed layouts, the GNN-based model is seemingly *more* vulnerable to our proposed method with degradations in NRMS and SSIM of up to 427.3% and 48.11% for a budget of 5% of cells.

4.3 Improving robustness of congestion predictors via momentum-based PGD

A number of techniques [3, 14, 19] have been proposed to mitigate the issue of robustness of deep networks, with some of the most reliable being certified defenses [6] and methods based on the principle of adversarial training [14]. In this work, we forgo investigating provable defenses and instead stick with PGD-based adversarial training. More concretely, each iteration of SGD, a portion of each batch (i.e. 50%) is perturbed via our method. Running PGD during training is expensive. One may exploit the Fast-FGSM algorithm proposed in [19], which demonstrates that by simply introducing random initialization points from which to compute adversarial perturbations, one projected gradient step is as effective as repeated steps during training while being significantly more efficient.

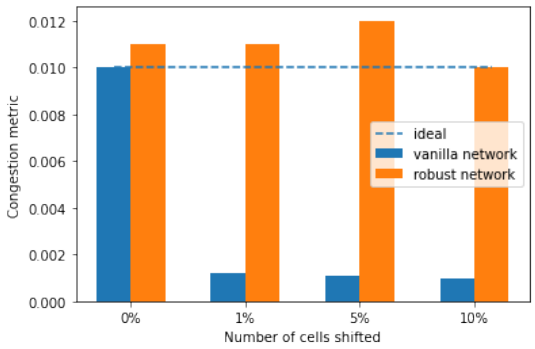


Figure 6: Robust congestion prediction statistic ($\frac{1}{HW}||f_\theta(M)||_F^2$) and percentage of cells that are allowed to move (looseness of ϵ_0 constraint).

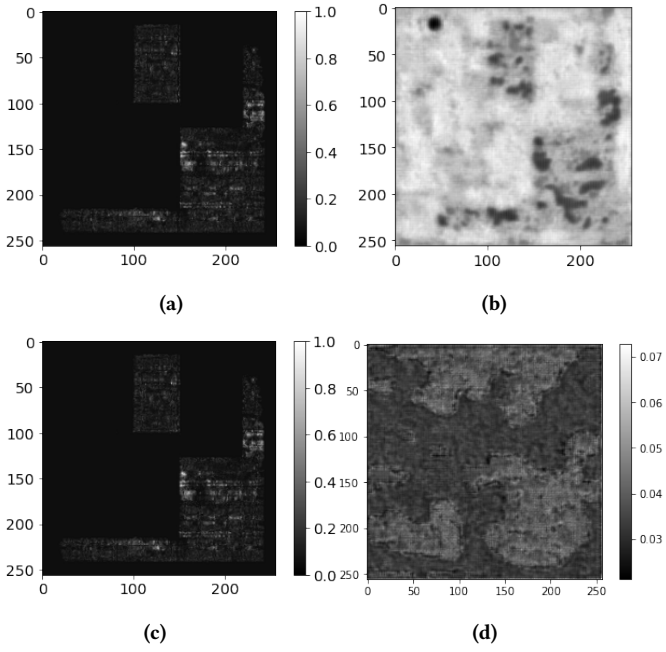


Figure 7: (a—b): Predictions for robust and nonrobust estimators (supervised loss). Both plots share the same gradient scale. (c—d) Predictions for robust and nonrobust estimators (unsupervised loss). Note the difference in gradient scale.

Using standard PGD, we train models on using the CircuitNet split and report the *robust test statistics* in the left column of Table 1. We see that models trained using adversarial training are *significantly* more robust with respect to both unsupervised and supervised perturbations. More concretely, we observe recovery of predictive quality primarily with respect to the metric driving perturbations (i.e. when predictors are trained to be robust to the unsupervised congestion metric, robustness is improved across all metrics, but most significantly for unsupervised congestion). In Figure 6, we provide a comparison between a vanilla network and a robust network trained via PGD. On the x-axis, we plot the ϵ_0 constraint, the percentage of cells that are free to be adjusted. On the y-axis, we plot the mean congestion metric $\frac{1}{HW}||f_\theta(M)||_F^2$ across the validation set. Note that the robust network maintains good performance, even as the number of cells increases.

The congestion value 0.01 is from a baseline predictor—the blue bar at 0% cells shifted. Note that the value may not be representative of prediction quality (instead, see Figure 4). Ideally bars should be the same height across perturbations. However, the vanilla model's predictions on perturbed layouts change significantly. In contrast, robust predictors generalize (orange bars have similar height).

5 CONCLUSION AND FUTURE WORK

In this paper, we have demonstrated that CNN-based congestion detection models are vulnerable to small perturbations. We have proposed and evaluated layout perturbations that are *guaranteed* to not alter a early global routing. To address these issues, we have proposed to apply adversarial training, demonstrating that such methods can improve robustness and generalization of deep learning-based EDA systems. The implication of our work is that designers should carefully evaluate deep learning-based models when employing ML-based CAD systems in EDA pipelines. More broadly, we hope that our perturbation methodology for evaluating vulnerabilities in congestion prediction and our adversarial training approach to make congestion predictions more robust can be adapted for evaluating ML-based EDA predictors.

ACKNOWLEDGMENTS

We acknowledge support from NSF CCF-2110419.

REFERENCES

[1] Mohamed Baker Alawieh, Wuxi Li, Yibo Lin, Love Singhal, Mahesh A. Iyer, and David Z. Pan. 2020. High-Definition Routing Congestion Prediction for Large-Scale FPGAs. In *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. 26–31. https://doi.org/10.1109/ASP-DAC47756.2020.9045178

[2] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press, USA.

[3] N. Carlini and D. Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy (SP)*. 39–57.

[4] Zhuomin Chai, Yuxiang Zhao, Yibo Lin, Wei Liu, Runsheng Wang, and Ru Huang. 2022. CircuitNet: An Open-Source Dataset for Machine Learning Applications in Electronic Design Automation (EDA). *SCIENCE CHINA Information Sciences* 65, 12 (2022), 227401–.

[5] Vidya A. Chhabria, Yanqing Zhang, Haoxing Ren, Ben Keller, Brucek Khailany, and Sachin S. Sapatnekar. 2021. MAVIREC: ML-Aided Vectored IR-Drop Estimation and Classification. *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (2021), 1825–1828.

[6] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. 2019. Certified Adversarial Robustness via Randomized Smoothing *(Proceedings of Machine Learning Research, Vol. 97)*. PMLR, Long Beach, California, USA, 1310–1320.

[7] Francesco Croce and Matthias Hein. 2019. Sparse and Imperceivable Adversarial Attacks. In *ICCV*.

[8] Francesco Croce and Matthias Hein. 2020. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*.

[9] Francesco Croce and Matthias Hein. 2021. Mind the box: l_1-APGD for sparse adversarial attacks on image classifiers. In *ICML*.

[10] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations*.

[11] Joseph L. Greathouse and Gabriel H. Loh. 2018. Machine Learning for Performance and Power Modeling of Heterogeneous Systems. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–6. https://doi.org/10.1145/3240765.3243484

[12] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

[13] Siting Liu, Qi Sun, Peiyu Liao, Yibo Lin, and Bei Yu. 2021. Global Placement with Deep Learning-Enabled Explicit Routability Optimization. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 1821–1824.

[14] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations (ICLR)*.

[15] Samuel K. Moore. 2018. DARPA Picks Its First Set of Winners in Electronics Resurgence Initiative. https://spectrum.ieee.org/tech-talk/semiconductors/design/darpa-picks-its-first-set-of-winners-in-electronics-resurgence-initiative

[16] Peter Spindler and Frank M. Johannes. 2007. Fast and Accurate Routing Demand Estimation for Efficient Routability-driven Placement. In *2007 Design, Automation & Test in Europe Conference & Exhibition*. 1–6. https://doi.org/10.1109/DATE.2007.364463

[17] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. 2017. One pixel attack for fooling deep neural networks. *CoRR* abs/1710.08864 (2017). arXiv:1710.08864

[18] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. *arXiv* abs/1312.6199 (2014).

[19] Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. 2020. Improving Adversarial Robustness Requires Revisiting Misclassified Examples. In *International Conference on Learning Representations*.

[20] Zhiyao Xie, Yu-Hung Huang, Guan-Qi Fang, Haoxing Ren, Shao-Yun Fang, Yiran Chen, and Jiang Hu. 2018. RouteNet: Routability prediction for Mixed-Size Designs Using Convolutional Neural Network. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–8.

[21] Cunxi Yu, Houping Xiao, and Giovanni De Micheli. 2018. Developing Synthesis Flows without Human Knowledge. In *Proceedings of the 55th Annual Design Automation Conference* (San Francisco, California) *(DAC '18)*. Association for Computing Machinery, New York, NY, USA, Article 50, 6 pages. https://doi.org/10.1145/3195970.3196026

[22] Yen-Ting Yu, Ya-Chung Chan, Subarna Sinha, Iris Hui-Ru Jiang, and Charles Chiang. 2012. Accurate process-hotspot detection using critical design rule extraction. In *DAC Design Automation Conference 2012*. 1163–1168.