Algorithm Selection Framework for Legalization Using Deep Convolutional Neural Networks and Transfer Learning

Renan Netto, Sheiny Fabre, Tiago Augusto Fontana, Vinicius Livramento, Laércio Lima Pilla, Laleh Behjat, Jose Luis Guntzel

Algorithm Selection Framework for Legalization Using Deep Convolutional Neural Networks and Transfer Learning *

Renan Netto[1], Sheiny Fabre[1], Tiago Augusto Fontana[1], Vinicius Livramento[1], Laércio L. Pilla[2,3], Laleh Behjat[4], and José Luís Güntzel[1]

[1]Embedded Computing Lab, Dept. of Computer Science and Statistics (INE-PPGCC), Federal University of Santa Catarina, Brazil – email: {renan.netto,sheiny.fabre,tiago.fontana, vinicius.livramento}@posgrad.ufsc.br, j.guntzel@ufsc.br
[2]Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400, Talence, France – email: laercio.lima-pilla@labri.fr
[3]INRIA, LaBRI, UMR 5800, F-33400, Talence, France
[4]University of Calgary, Calgary, AB T2N 1N4, Canada – email: laleh@ucalgary.ca

Abstract

Machine learning models have been used to improve the quality of different physical design steps, such as timing analysis, clock tree synthesis and routing. However, so far very few works have addressed the problem of algorithm selection during physical design, which can drastically reduce the computational effort of some steps. This work proposes a legalization algorithm selection framework using deep convolutional neural networks. To extract features, we used snapshots of circuit placements and used transfer learning to train the models using pre-trained weights of the Squeezenet architecture. By doing so we can greatly reduce the training time and required data even though the pre-trained weights come from a different problem. We performed extensive experimental analysis of machine learning models, providing details on how we chose the parameters of our model, such as convolutional neural network architecture, learning rate and number of epochs. We evaluated the proposed framework by training a model to select between different legalization algorithms according to cell displacement and wirelength variation. The trained models achieved an average F-score of 0.98 when predicting cell displacement and 0.83 when predicting wirelength variation. When integrated into the physical design flow, the cell displacement model achieved the best results on 15 out of 16 designs, while the wirelength variation model achieved that for 10 out of 16 designs, being better than any individual legalization algorithm. Finally, using the proposed machine learning model for algorithm selection resulted in a speedup of up to 10x compared to running all the algorithms separately.

1 Introduction

Machine Learning (ML) models have been used in physical design for predicting different metrics, such as the result of clock tree synthesis (CTS) algorithms [20], circuit timing [14, 21, 22, 3] and

*Author's accepted version. Definitive version available at https://doi.org/10.1109/TCAD.2021.3079126

routing violations [37, 5, 32, 34, 35, 38, 11, 27, 36, 18]. The aim of these algorithms has been to improve the quality of a given synthesis or optimization step and/or speeding up execution. The predicted metrics may be used to anticipate the result of a given step, thus allowing for (1) guiding an earlier optimization step to reduce the impact on the latter steps, (2) choosing the most appropriate parameters to configure the employed algorithm or (3) selecting between the available algorithms the one that leads to the best quality results. The selection between the algorithms is usually referred to as **algorithm selection**.

One of the steps where algorithm selection can be important is legalization. The complexities of modern circuits force legalization algorithms to handle complex design challenges, such as physical floorplan complexity, routability issues and presence of multi-row cells. Therefore, it is hard to choose a single algorithm that outperforms others in every metric and for all circuit types. Different algorithms perform better in different scenarios. On the other hand, running different legalization algorithms just to be able to choose the best one for a given circuit can lead to overly long execution times, especially for incremental placement where multiple candidate solutions are tried. In this work, we have developed a framework to select a legalization algorithm that best fits a given placement, thus increasing the quality of the outcome and avoiding prohibitively long execution times. The main contributions of this work are:

- A Convolutional Neural Network (CNN)-based algorithm selection model for legalization. Since CNNs use only images as input features, the proposed model can use the same set of features to predict different metrics.
- Development of a feature extraction method for legalization that is independent of the metrics being predicted.
- Extensive experimentation and analysis for training deep CNN using transfer learning, including which pre-trained architecture to use.
- Experimental validation of the proposed algorithm selection framework using state-of-the-art legalization algorithms.

The remaining sections are organized as follows. In Section 2 the related work and background are discussed. Section 3 presents the proposed algorithm selection framework, providing details on feature extraction, CNN training process and the framework integration with the physical design flow. In Section 4, the experimental results are given. Finally, concluding remarks are provided in Section 5.

2 Related work and Background

2.1 Legalization Problem

Legalization is a step in the placement stage where cells are slightly moved to be in legal locations, i.e. no overlap, in rows and aligned with power rails. In multi-row legalization, the cells may have different heights. Figure 1 (a) shows an example of the input to a legalization problem is shown. In this figure, cells are shown as blue rectangles and a fixed macroblock appears as a gray rectangle. A legalization algorithm must produce a legal placement as the one in Figure 1 (b) while moving the cells as least as possible.

A legal placement can be produced by using greedy heuristics [7, 39, 9], dynamic programming [33], Integer Linear Programming (ILP) [17] or modeling the problem as a Linear Complementarity Problem (LCP) [6, 26]. During legalization, it is important to observe the spatial characteristics of the problem. For example, when a large group of cells are close to each other, it

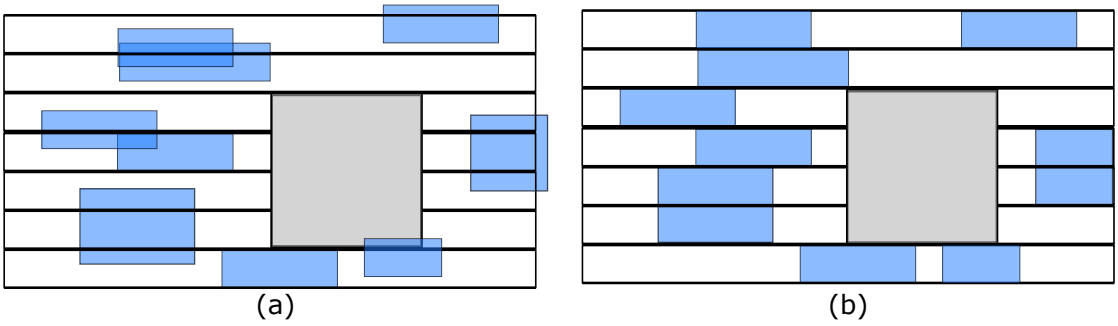

Figure 1: Example of the legalization problem. Blue rectangles represent circuit cells, while the gray rectangle represents a macroblock. (a) Input of the problem. (b) Possible solution of a legalization algorithm.

becomes more difficult for the algorithm to legalize the circuit without moving many cells. CNN models achieve good results with spatial data because they identify patterns in objects that are spatially close to each other in the input data. When predicting the outcome of legalization algorithms, the model needs to identify patterns in the cells locations and, therefore, we claim that CNN is the appropriate model for that.

2.2 Machine Learning in Physical Design Applications

Table 1 presents a summary of the related work using ML grouped by their targeted physical design step, alongside the type of ML model used. These works can be grouped into four categories according to the physical design step they are intended to: clock tree networks, timing analysis, routing and legalization.

Kahng et al. have presented one of the first works to use ML models in physical design [20]. In this work, trained regression models are developed to predict the outcome of CTS engines. These models are used for two different purposes: choosing between two CTS engines for a given set of parameters and choosing the best set of parameters for a single CTS engine.

There are a number of papers using ML for timing analysis. Han et al. [14] use regression models to fix miscorrelations between different commercial signoff timing engines and between a signoff tool and a commercial design implementation tool. The work in [21] focuses on predicting timing in signal integrity (SI) mode based on timing reports from non-SI mode. In [22] random forests and regression trees were used to predict path-based slack from graph-based timing analysis. In [3] different non-convolutional models are used to predict signoff timing from the circuit features.

Several works explored ML models to prevent routing violations [37, 5, 32, 34, 35, 38, 11, 27, 36, 18] using different strategies. In [37] and [5] circuit layout features such as pin distribution and density parameters are extracted to train non-convolutional models. The main difference between these two works is that [37] identifies only the number of detailed routing violations, whereas [5] also finds their locations.

Other works on routing explored convolutional models. In [32], [34] and [18] placement and global routing features are used to predict where routing violations will happen. In [35, 36], a model that identifies pin patterns that would likely lead to violations is trained. In [38] and [27], circuit placement is used to generate violation hotspot maps. In [11], a reinforcement learning approach is developed. This approach is based on AlphaGo Zero and models the routing problem

Table 1: Summary of related work on the use of machine learning for physical design applications

Physical design step	Work	ML model
CTS	Kahng et al. (2013) [20]	non-convolutional
Timing analysis	Han et al. (2014) [14] Kahng et al. (2015) [21] Kahng et al. (2018) [22] Barboza et al. (2019) [3]	non-convolutional
Routing	Zhou et al. (2015) [37] Chan et al. (2017) [5]	non-convolutional
	Tabrizi et al. (2018) [32] Xie et al. (2018) [34] Yu et al. (2019) [35] Zhou et al. (2019) [38] Gandhi et al. (2019) [11] Liang et al. (2020) [27] Yu et al. (2020) [36] Hung et al. (2020) [18]	convolutional
Legalization	Netto et al. (2019) [28]	non-convolutional, convolutional
	This work	convolutional

as a two-player game, where one player tries to route the circuit and the other one tries to remove the violations.

The only related work that focuses on predicting the quality of legalization algorithms is our previous work [28]. In that work, we trained an ML model to identify circuit regions that would result in large displacement after legalization. Then, the trained model was used as a pruning mechanism in a circuit partitioning strategy, in order to avoid partitions which would lead to large displacement. Since legalization is called many times during the optimization process, improving the legalization solution consequently improves the quality of those optimizations.

At this point, it is worth noting four characteristics of the related work: (1) most of the latest works use convolutional models since they use spatial features such as cell and interconnection locations that seem to be captured properly by those models; (2) only Kahng et al. [20] make use of ML to perform algorithm selection in a physical design flow. However, since their work focuses specifically on CTS, and uses only clock tree features to train the models, their methodology cannot be easily adapted to other physical design steps. Algorithm selection has been successfully applied to different application domains, such as in [13, 31, 25, 23], and can be used in other physical design steps as well; (3) most related work target timing analysis and routing, while only a few address other steps like CTS and legalization, leaving much room for investigating the application of ML to other physical design steps; (4) no related work explored transfer learning to use pre-trained weights and hence, reduce the training runtime. A shorter training time allows a wider parameter exploration, as it takes shorter time to evaluate different configurations.

2.3 Supervised learning concepts

Supervised machine learning (SML) is the process of mapping a set of features, X, to a set of labels, Y. In SML, a function $f : X \to Y$ that maps each feature $x \in X$ to a label $y \in Y$ as accurately as possible is determined [29]. To that purpose, data is divided into two sets, the **training set** and **the validation set**. The training set is used to develop the model and the validation set is used to examine its efficacy.

Different models can be used for SML, such as: Decision Trees, Artificial Neural Networks, Support Vector Machines, Random Forests, and Convolutional Neural Networks (CNNs). CNNs are capable of identifying patterns in images, making them suitable for problems that have spatial data [4, 12] such as the legalization problem.

In CNN, the training is done using the back propagation method which initializes the network weights, then feeds the model with training data adjusting the weights to fit the data to their corresponding labels [12]. Since CNNs typically have several hidden layers (constituting a deep learning model), training them requires a large amount of data and time. Two techniques can be used to reduce the complexity of the training process: **transfer learning** and **data augmentation**. Transfer learning consists in using weights from a previous training process instead of initializing the network with random weights. This reduces the training time, as it is not necessary to train the network from scratch, but rather adjust the weights to fit the current data. Data augmentation aims to improve the model quality without requiring a large amount of training data. Since convolutional models use image data, it is possible to apply image transformations (such as scaling or rotation) on the available data to artificially generate more data, improving the model quality. Data augmentation is also useful in handling imbalanced data sets where the number of positive and negative instances are vastly different, as it enables the generation of more data of a given underrepresented class.

To evaluate the quality of a model, several metrics are available. The **confusion matrix** is a table that presents the performance of the model with four metrics:

- True positive (TP): number of positive instances correctly classified as positive.
- True negative (TN): number of negative instances correctly classified as negative.
- False positive (FP): number of negative instances incorrectly classified as positive.
- False negative (FN): number of positive instances incorrectly classified as negative.

Given a confusion matrix, it is possible to measure more sophisticated metrics for models with imbalanced data [30]:

- **Accuracy (A)**, $\frac{TP+TN}{TP+TN+FP+FN}$: is the number of correctly classified samples divided by the total number of samples.
- **Precision (P)**, $\frac{TP}{TP+FP}$: is the number of positive classified samples divided by the number of positive samples.
- **Recall (R)**, $\frac{TP}{TP+FN}$: is the proportion of positives identified correctly.
- **F-score**, $2 \times \frac{P \times R}{P+R}$: is the harmonic mean of precision and recall.

Two other important concepts in SML are **over-fitting** and **under-fitting**. These are concepts used to evaluate how well the model fits the data [4]. A model is under-fitting the data when it has a high error on the training data. This typically means that the model is too simple to fit the data. Under-fitting can usually be solved by increasing the model complexity. A model

is over-fitting the data when it has high error on the validation data, although presenting low training error. This means the model is too biased to the training data, thus it is not able to generalize to instances that were not seen during training. There are different ways of avoiding over-fitting such as increasing the amount of training data (possibly through data augmentation), adding dropout layers (when using neural networks), or using regularization technique [10].

3 Proposed Algorithm Selection Framework

In this work, we apply SML to the outcomes of the legalization during placement. Our proposed model is a classification problem where a list of legalization algorithms $\mathcal{L}$, a placement defined by a set of cells C and their locations, and a set of nets N are given. The output of the model is an integer variable $y \in \{1, 2, ..., |\mathcal{L}|\}$, indicating which algorithm results in the best legalization for a specific metric. The goal of the model is to select the algorithm that results in the best solution without having to run all legalization algorithms, thus saving execution time.

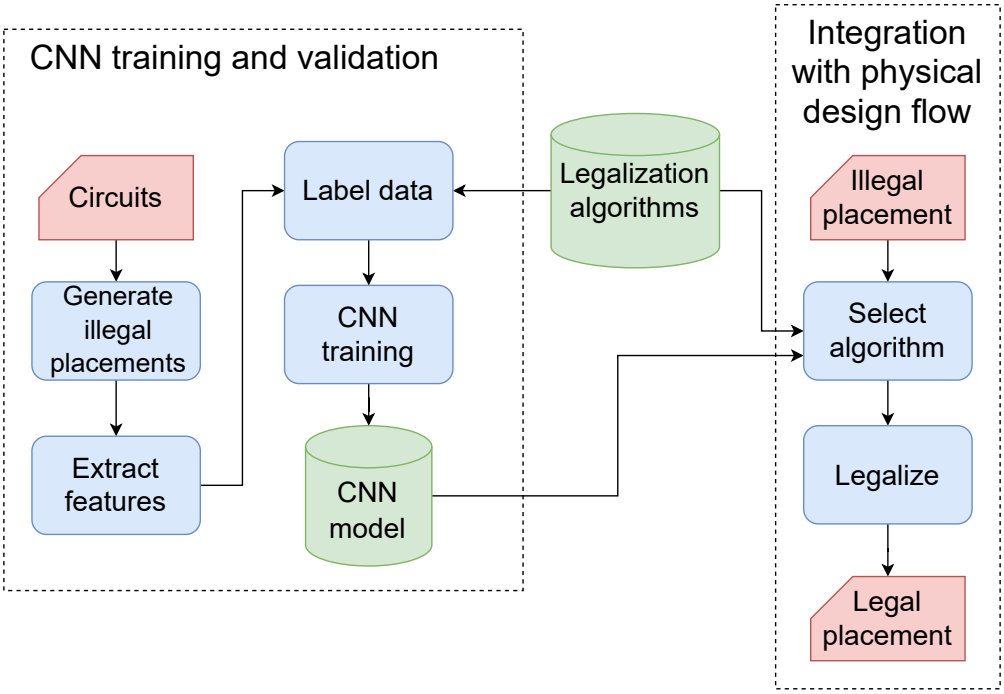

Figure 2: Flowchart of the proposed algorithm selection framework.

Figure 2 shows the flowchart of our proposed algorithm selection framework, which is made of two main parts: CNN training and validation, and integration with the physical design flow. The former part has a few steps. First, we generate illegal placements from the input circuits to increase the amount of training and validation data. We extract features from the training data by saving the images of each placement and we label the data according to the score achieved by each legalization algorithm. Given that, we train the CNN model. The whole training process is executed outside the physical design flow, and the CNN model is saved. The algorithm selection step itself is integrated in the physical design flow. In this step we use the CNN model to predict the best legalization algorithm to use for each illegal placement. Then, we legalize the placement

using the selected algorithm.

3.1 Proposed data generation strategy

Algorithm 1 presents the proposed data generation strategy to train and validate the CNN model. This algorithm corresponds to steps "Generate illegal placements" and "Extract features"

Algorithm 1: GENERATE_DATA(C, N, $\mathcal{L}_i$, R)

Input : Set of cells C, set of nets N, legalization algorithm $\mathcal{L}_i$ and circuit region R
Output: Training data

```
1  for c_i ∈ C do
2  |  l'(c_i) ← l(c_i);
3  end
4  n_samples ← 1000, F ← ∅;
5  for num_sample ← 1 to n_samples do
6  |  MOVE_CELLS(C, R);
7  |  I ← SAVE_IMAGE(C, N, R);
8  |  Λ_i, result_i ← L_i(C, N, R);
9  |  δ_i ← EVALUATE_SOLUTION(Λ_i, C, N);
10 |  F ← F ∪ (I, δ_i, result_i);
11 |  for c_i ∈ C do
12 |  |  l(c_i) ← l'(c_i);
13 |  end
14 end
15 return F;
```

from Figure 2. It receives as input the placement information (C and N), a set of legalization algorithms $\mathcal{L}_i$, and the rectangular region of the entire circuit $R = (X_{left}, X_{right}, Y_{top}, Y_{bottom})$. The cell c_i information includes a location $l(c_i) = \{x(c_i), y(c_i)\}$ which can be illegal, and the cell dimensions $d(c_i) = \{w(c_i), h(c_i)\}$. In the end, the algorithm outputs data from illegal placements to train the CNN model.

The algorithm starts by saving the initial locations of the cells (lines 1–3) and specifying the number of samples to generate for the circuit (line 4). To generate each sample, we apply a perturbation to the circuit placement by moving the cells (line 6). Details on the cell movement will be given in Algorithm 2. After perturbing the circuit, we save the placement image with the new locations (line 7). Then, the legalization algorithm $\mathcal{L}_i$ is used to find legal locations to the perturbed placement (line 8). It is worth remarking that the legalization algorithm only returns the legal locations for the cells, it does not actually move the cells. With the legal locations, the solution is evaluated and both the placement image I and the legalization result are added to the set of input features $\mathcal{F}$ (lines 9–10). Then, the cells are moved back to their original locations, so that they can be moved again in the next iteration (lines 11–13). At the end, Algorithm 1 returns the set of features for this specific circuit and legalization algorithm. This process is repeated for all the circuits and all the legalization algorithms used to train the CNN model.

It is important to note a few aspects of Algorithm 1. The first one is that it is possible that the legalization algorithm fails to legalize a given placement. Hence, the return of the legalization algorithm can be the outcome of the legalization algorithm (if it was successful or not). This outcome is also saved in the data features, since it is important to know when the legalization failed. The second observation is that the EVALUATE_SOLUTION function is generic, and can be used to evaluate different metrics. Due to the flexibility of CNN models, we can use the same set of features to classify data according to different metrics, as long as the metric is related to

the locations of cells and nets. Therefore, the EVALUATE_SOLUTION function can be changed to evaluate the metric the designer is more interested in, such as cell displacement, wirelength, density, or timing.

The MOVE_CELLS function, which is responsible for perturbing the placement by moving the cells on each sample iteration, is shown in Algorithm 2. It receives as input the set of cells

Algorithm 2: MOVE_CELLS(C, R)

Input : Set of cells C and circuit region R
Output: Set of moved cells C

```
1  foreach c_i ∈ C do
2      if IS_MOVABLE(c_i) then
3          r_x ←RANDOM(-10000, 10000);
4          r_y ←RANDOM(-10000, 10000);
5          x(c_i) ← x(c_i) + r_x;
6          y(c_i) ← y(c_i) + r_y;
7          x(c_i) ←min(X_right − w(c_i), max(X_left, x(c_i)));
8          y(c_i) ←min(Y_top − h(c_i), max(Y_bottom, y(c_i)));
9      end
10 end
```

C and the circuit region R, and randomly moves the movable cells in the circuit. Some cells, such as blockages, are fixed and cannot be moved while generating a new illegal placement. The Algorithm starts by picking a cell and checking if it is movable (line 2). If the cell is movable, the function generates a random amount of movement in x and y directions (lines 3–4). The movements range from $-10,000$ to $10,000$ placement units. The cell is moved to the new location while ensuring it remains inside the circuit bounds (lines 5–8). The benchmarks evaluated in this work have dimensions ranging from 342,000 to 900,000 placement units. Hence, the amount of random movement applied to each cell represents at most only 3% of the circuit dimensions, resulting in a small perturbations of the initial placement. We believe that this amount of movement is enough to generate placements with diverse characteristics to put in evidence the efficacy of the distinct legalization algorithms, without being too different from the original global placement, as the movement is not big enough to separate clustered regions.

3.2 Feature extraction

Features with spatial properties are better suited for CNN models. Therefore, we generate snapshot images of the circuit illegal placements. However, we need to generate the images in a way that the model can identify four essential features on them: (1) the difference between fixed and movable cells; (2) the difference between cells in different fence regions; (3) the existence of cell overlaps; (4) the nets connecting the cells. Therefore, our images are generated as follows: (1) fixed cells are gray; (2) movable cells have colors according to which fence region they belong to; (3) all objects have lower opacity, so that it is possible to detect overlaps between them; (4) The nets are black to differentiate from cells. Since the circuit was not routed yet, we represent the nets by their Steiner trees. Once an image is generated, it is flipped in the x and y axes to quadruple the number of available images. Examples of generated images for a sample circuit are illustrated in Figure 3. Observe that, even though the circuit has a lot of nets, many local nets are too short to be visible in the image. For the benchmarks that we use in this work, each pixel can cover up to 9 sites. If the net length is smaller than this, it will not be visible in the image. However, we observed that the CNN could still extract useful information from the visible nets.

We further investigate the impact of the image size in the CNN accuracy in Section 4.4.

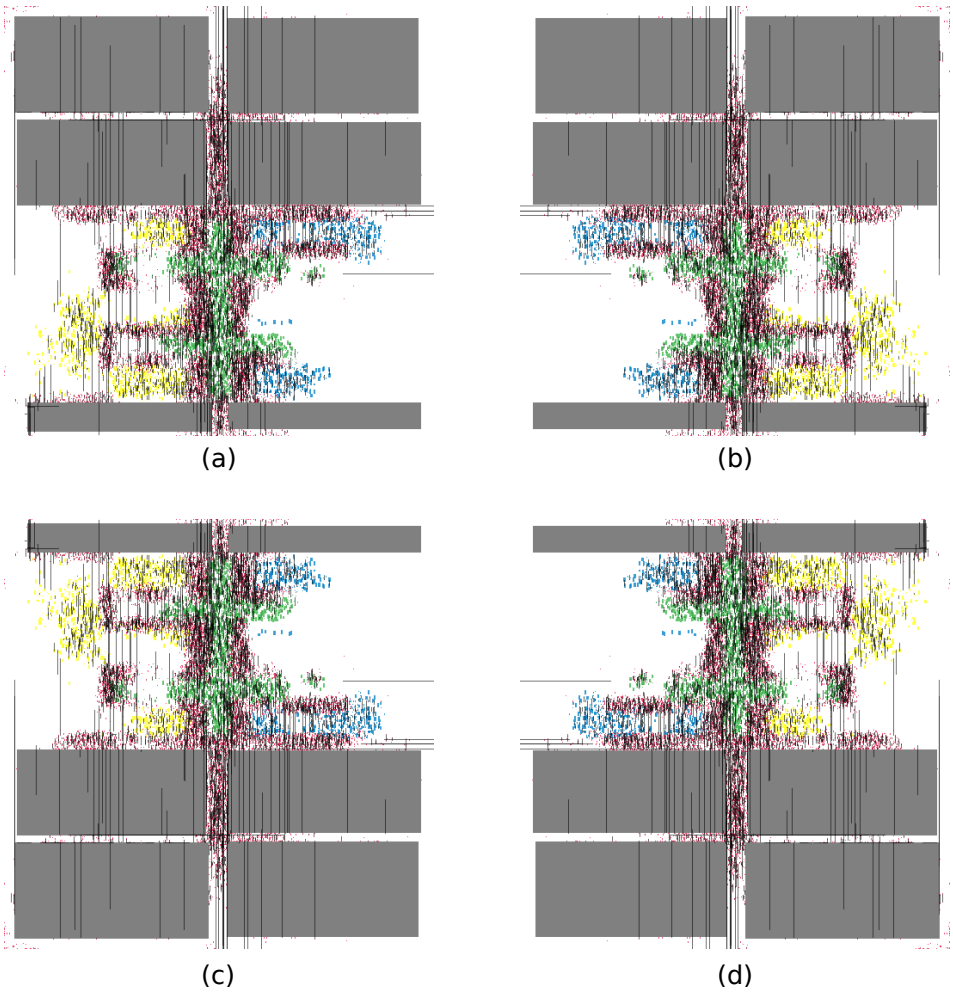

Figure 3: Examples of images of a circuit. (a) Original image. (b) image flipped in the y axis. (c) image flipped in the x axis. (d) image flipped in x and y axes.

Each image must also be labeled based on the results of the legalization. Then, before training the model, an additional labeling step is executed where we iterate through all the images, and add labels to tell the CNN model which legalization algorithm achieved the best performance for the metric that is under evaluation. This allows us to train models for different metrics without having to generate different images for each model. For example, if we want to train CNN models for cell displacement and circuit wirelength, we just need to change the function EVALUATE_SOLUTION in Algorithm 1 to measure each metric and use the same placement images for both of them. Then, the next step for feature extraction is the same as for any legalization algorithms, since the CNN model is independent of the semantics of the metric being predicted.

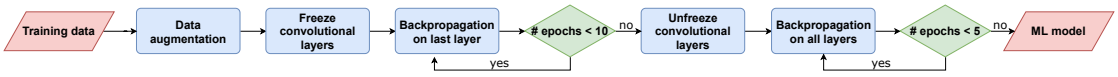

Figure 4: Flowchart of the proposed CNN training process.

3.3 Convolutional neural network training process

In order to reduce the training time, we use a transfer learning model with pre-trained weights. The available CNN models were usually pre-trained using the ImageNet dataset, which contains over 14 million images. We chose the SqueezeNet model [19] as it presented the best results on our experimental evaluation (more details are given in Section 4.2). In addition, we applied the data augmentation method for training the model, as described in Section 2.3. The data augmentation process consists in applying transformations on the images generated by Algorithm 1 to artificially generate more data. This way, we can reduce over-fitting during the training process.

Figure 4 shows the flowchart of the training process we used for the CNN models. The first step is data augmentation. We performed the following transformations on the images: (1) flipping the image horizontally and vertically; (2) applying image zoom of up to 5% (the exact percentage is chosen randomly for each image). We did not apply rotation and image distortions, as those transformations will generate images that are not realistic in the frame of circuit legalization. Figure 3 shows examples of these transformations.

The augmented data is then used to provide batches of the model in the back-propagation process. The training process is then divided into two stages. In the first stage, we freeze the weights of all convolutional layers and adjust only the weights of the last (fully connected) layer. This first step was executed for 10 epochs with a learning rate of 0.01. In the second stage, we unfreeze the convolutional layers and adjust all weights for 5 more epochs. This is necessary because our training data is very different from the ImageNet benchmarks, so we need to fine-tune the weights of the convolutional layers to reflect the differences. However, the early layers of the model identify more general patterns that are not specific to the image semantics, so we only need to perform small changes on those weights. For this reason, we employ different learning rates for different sections of the neural network, as follows: we equally divide the neural network layers into sections, then we use learning rates ranging from 0.00001 to 0.002 on those sections. This strategy is called **discriminative fine tuning**. It allows for a better tuning in the latter layers, where the model becomes more specialized for our problem [15]. Details of how we chose the training parameters are provided in Section 4.3.

3.4 Integration with the physical design flow

After training and validating the CNN model, we integrated it in the legalization flow. Given an illegal placement, we use the CNN model to select which algorithm is the best one to legalize it.

The proposed CNN-based algorithm selector runs only one of the legalization algorithms, as described in Algorithm 3. The CNN algorithm selection receives as input the set of cells and nets C and N, and a CNN model $\mathcal{M}$, which will select the best legalization algorithm $\mathcal{L}_{best} \in \mathcal{L}$ (line 1). Based on the CNN model output, it executes the appropriate legalization algorithm (line 2) and moves the cells to their legal locations found by $\mathcal{L}_{best}$ (line 4–6) if the placement was successfully legalized. It is possible that the chosen algorithm or all algorithms fail to legalize the placement. In this case, the placement needs to be redone. However, in our experiments this did not happen.

In order to evaluate the efficiency and impact of our algorithm selection strategy (CNN), we compare it to a reference version (LEG) that selects the best algorithm by actually executing all

Algorithm 3: ALGORITHM_SELECTOR-CNN (C, N, $\mathcal{M}$)

Input : Set of cells C, set of nets N and CNN model $\mathcal{M}$
Output: Legalized circuit

```
1  L_best ← M(C, N);
2  Λ_best, result_best ← L_best(C, N);
3  if result_best then
4  |   foreach c_i ∈ C do
5  |   |   l(c_i) ← λ_best(c_i);
6  |   end
7  end
```

the legalization algorithms and comparing the results. In Algorithm 4, the implementation of the LEG selector is given.

Algorithm 4: ALGORITHM_SELECTOR-LEG (C, N, $\mathcal{L}$)

Input : Set of cells C, set of nets N, circuit region R and list of legalization algorithms $\mathcal{L}$
Output: Legalized circuit

```
1  δ_min ← ∞;
2  Λ_best ← ∅;
3  foreach L_i ∈ L do
4  |   Λ_i, result_i ← L_i(C, N, R);
5  |   if result_i then
6  |   |   δ_i ← EVALUATE_SOLUTION(Λ_i, C, N);
7  |   |   if δ_i < δ_min then
8  |   |   |   δ_min ← δ_i;
9  |   |   |   Λ_best ← Λ_i;
10 |   |   end
11 |   end
12 end
13 if Λ_best ≠ ∅ then
14 |   foreach c_i ∈ C do
15 |   |   l(c_i) ← Λ_best(c_i);
16 |   end
17 end
```

The LEG selector receives as input a set of cells C, a set of nets N, circuit region R and a set of legalization algorithms $\mathcal{L}$. It iterates through $\mathcal{L}$ to select the best one (lines 3–12) based on a specific performance measure. For each legalization algorithm, it legalizes the placement (line 4), evaluates the quality (line 6) and updates the best one so far (lines 7–10). As in line 8 of Algorithm 1, the legalization algorithm does not actually move the cells to the legal locations. Instead, it returns a set of legal locations Λ_i and a Boolean variable $result_i$ specifying if the placement was successfully legalized. The legal locations Λ_i are used to evaluate the solution quality, and if the quality (δ_i) is better than the best solution, they are saved. After evaluating all the legalization algorithms, cells are moved to the best locations saved in Λ_{best} (lines 13–17).

Similarly to Algorithm 1, the EVALUATE_SOLUTION function (line 6) may compute any desired quality metric. Therefore, it is possible to use Algorithm 4 for different metrics or even combining multiple metrics by only changing one function. In this work, we used two metrics for solution quality: **cell displacement** and **wirelength variation**.

It is important to highlight that, while the LEG selector needs to run all the legalization algorithms, the CNN-based selector needs to run one algorithm, and the CNN model inference. As a consequence, the complexity of Algorithm 4 is proportional to the number of legalization algorithms available, while the complexity of Algorithm 3 is proportional, in the worst case, to the slowest legalization algorithm. Even though the CNN-based selector still needs to run the CNN model, in Section 4 we show that the inference time is much shorter than the execution time of any legalization algorithm. Therefore, using CNN allows a speedup of the algorithm selection process roughly proportional to the number of legalization algorithms to evaluate.

4 Experimental evaluation

4.1 Experimental setup

4.1.1 Equipment and implementation details

For training the proposed CNN model, a CentOS workstation with an Intel® Xeon® Gold 6148 processor with 20 cores at 2.4 GHz, 750GB RAM and an NVIDIA Tesla V100 GPU was used. For the evaluation with the physical design flow, an Ubuntu 18.04 workstation with an Intel® Core® i7-3537U processor with 4 cores at 2.00 GHz and 6GB DDR3 1600 MHz RAM was used. We used the fast.ai 1.0.61 library [16] for training and the SqueezeNet architecture for transfer learning [19]. All experiments are available under public domain [2].

4.1.2 Benchmarks

The CNN model was trained and evaluated using ICCAD 2017 CAD Contest benchmarks [8]. The benchmark set is presented in Table 2 and consists of 16 circuits with sizes ranging from 29k to 130k cells. It is ideal for testing our framework as it includes challenges of advanced technology nodes, such as multi-row cells (up to 4 rows) and physical floorplan complexity such as fence regions and multiple macroblocks. The table also shows the area of each circuit in sites and rows as well as placement units, where each site width and each row height contains 200 and 2000 placement units, respectively. Given the number of sites, we can calculate how many sites are represented by each pixel[1] (shown in the rightmost column of the table).

4.1.3 Legalization algorithms

Among the legalization algorithms available in the literature, **HAO** [26], **ZIR** [39] and **ODP** (from OpenDP) [9] are the only ones adapted to the ICCAD 2017 benchmark. While all three bring improvements over previous works and incorporate routability and technology constraints to the problem formulation, none of them is clearly the best one for all the circuits in the benchmarks. Such particularity makes them a very interesting choice for our experiments. The binaries of the first two algorithms were provided by their respective authors, whereas OpenDP was compiled from its source code publicly available [1].

4.1.4 Evaluation metrics

In order to investigate if the proposed framework is robust enough to be useful for different physical design metrics, we trained models to classify the data using two different metrics, giving rise to two versions of CNN model: one that selects the legalization algorithm that results in the

[1] We used images with size of 500x500 pixels, as stated in Section 4.1.5. Hence, the number of sites per pixel is calculated as the number of sites divided by 500.

Table 2: Number of cells, nets, area and number of sites per pixel of each circuit in the ICCAD 2017 CAD Contest benchmark set.

Circuit	# cells	# nets	sites x rows	area (plac. units)	# sites/pixel
des_perf_b_md1	113K	113K	3K x 0.3K	600K x 600K	6
des_perf_b_md2	113K	113K	3K x 0.3K	600K x 600K	6
edit_dist_1_md1	131K	133K	3.61K x 0.36K	722K x 722K	7
edit_dist_a_md2	127K	131K	4K x 0.4K	800K x 800K	8
fft_2_md2	32K	33K	1.71K x 0.17K	342K x 342K	3
fft_a_md2	31K	32K	4K x 0.4K	800K x 800K	8
fft_a_md3	31K	32K	4K x 0.4K	800K x 800K	8
pci_bridge32_a_md1	30K	30K	2K x 0.2K	400K x 400K	4
des_perf_1	113K	113K	2.23K x 0.22K	445K x 445K	4
des_perf_a_md1	109K	110K	4.5K x 0.45K	900K x 900K	9
des_perf_a_md2	109K	110K	4.5K x 0.45K	900K x 900K	9
edit_dist_a_md3	127K	131K	4K x 0.4K	800K x 800K	8
pci_bridge32_a_md2	30K	30K	2K x 0.2K	400K x 400K	4
pci_bridge32_b_md1	29K	29K	4K x 0.4K	800K x 800K	8
pci_bridge32_b_md2	29K	29K	4K x 0.4K	800K x 800K	8
pci_bridge32_b_md3	29K	29K	4K x 0.4K	800K x 800K	8

lowest cell displacement for a given placement, and another that selects the algorithm resulting in the largest wirelength improvements (or the smallest wirelength degradation). Hereafter, these models are denoted by *Disp-CNN* and *WL-CNN*.

4.1.5 ML setup

To increase the number of available legalization samples, we generate random illegal placements for each circuit to increase the amount of input data. In our experiments, for each benchmark circuit, one thousand placements were generated[2], resulting in a total of 16 thousand placements. We used 13 circuits for training and the remaining 3 for validation, resulting in about 20% of the data being used for validation. We selected the validation circuits in a way that the three classes have a similar representation in the validation set. Since the distribution of classes among the circuits is different for each metric, we used a different validation set for each model. For Disp-CNN we used $des\_perf\_b\_md2$, $edit\_dist\_a\_md2$ and fft_2_md2 as the validation set. For WL-CNN we used $edit\_dist\_1\_md1$, fft_a_md3 and $des\_perf\_a\_md1$ as the validation set. By separating the circuits this way we can verify if the CNN model can generalize its prediction to circuits that were not seen during training. Finally, we re-scaled all images to have the same size (500×500 pixels), even though we are using circuits of different sizes.

Figures 5 and 6 show the distribution of the algorithms leading to the best results for each placement in terms of cell displacement and wirelength variation, respectively. Based on these figures, one can notice that there is no clear best algorithm, as the best algorithm changes not only from circuit to circuit but also for different placement instances of a given circuit and a given metric. For example, when using wirelength variation as the evaluation metric for fft_a_md2, about 50% of the placements are best legalized using HAO, but 30% of them should use ZIR and 20% should use ODP.

[2]Each placement was generated by applying random movements on the initial placement benchmarks provided by the ICCAD 2017 CAD Contest.

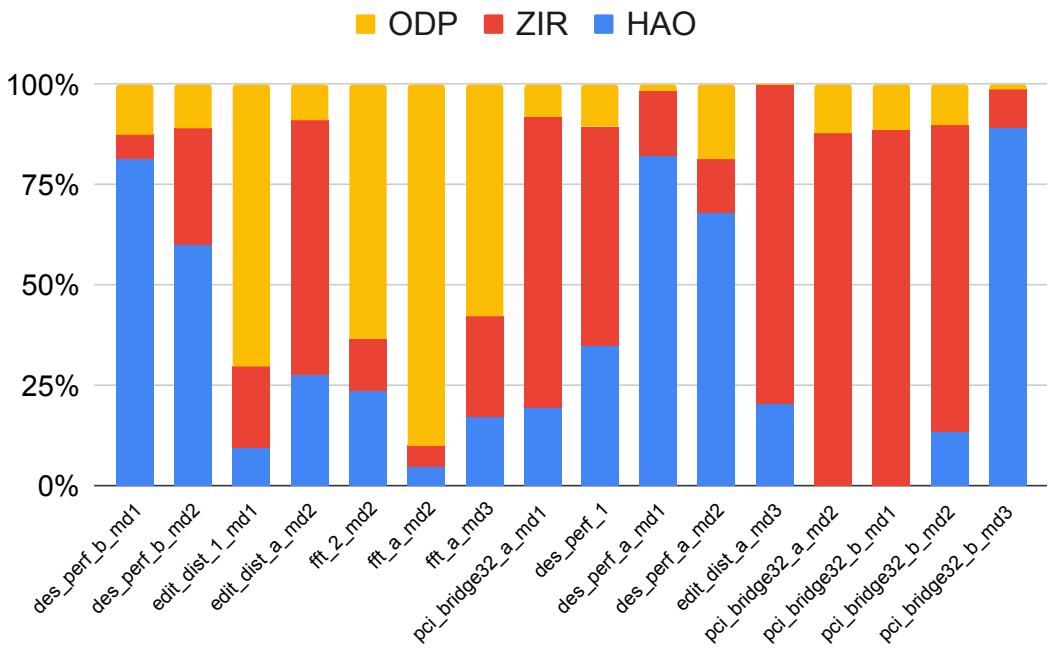

Figure 5: Distribution of best legalization algorithm in terms of cell displacement (y axis) for each placement generated from the benchmark set (x axis).

We also observed that when an algorithm is not the best option for a given placement, in most cases there is a relevant difference in quality when compared to the best solution. For both cell displacement and wirelength variation, when HAO was not the best, it was at least 7% worse than the best algorithm in more than half of the cases. When analyzing ZIR and ODP for both cell displacement and wirelength variation as well, we observe that they were at least 10% and 16% worse, respectively, than the best solution in more than half of the cases. Those results show that if we use only one of the three algorithms, we may lose in legalization quality.

4.2 CNN architecture selection

The first parameter that can impact the model's quality is the CNN architecture. The fast.ai library provides variations of the following architectures: AlexNet, DenseNet, ResNet, SqueezeNet, VGGNet. We evaluated only ResNet, SqueezeNet, and AlexNet because the other two architectures were too big to fit in the NVIDIA Tesla V100 GPU's memory. Table 3 shows the F-score (see Section 2.3) of each CNN architecture (columns) for each class (rows). We focus on the WL-CNN model because it showed the lowest F-scores, making it the main model to be optimized for the moment.

The three architectures in Table 3 reach similar F-scores on average. However, Squeezenet achieved a slightly better F-score for all three classes when compared to the other two architectures. In addition, the Squeezenet architecture requires 50x fewer parameters than AlexNet and is smaller than Resnet and therefore, has a smaller inference time [19]. Because of that, we chose SqueezeNet for the experiments in this work.

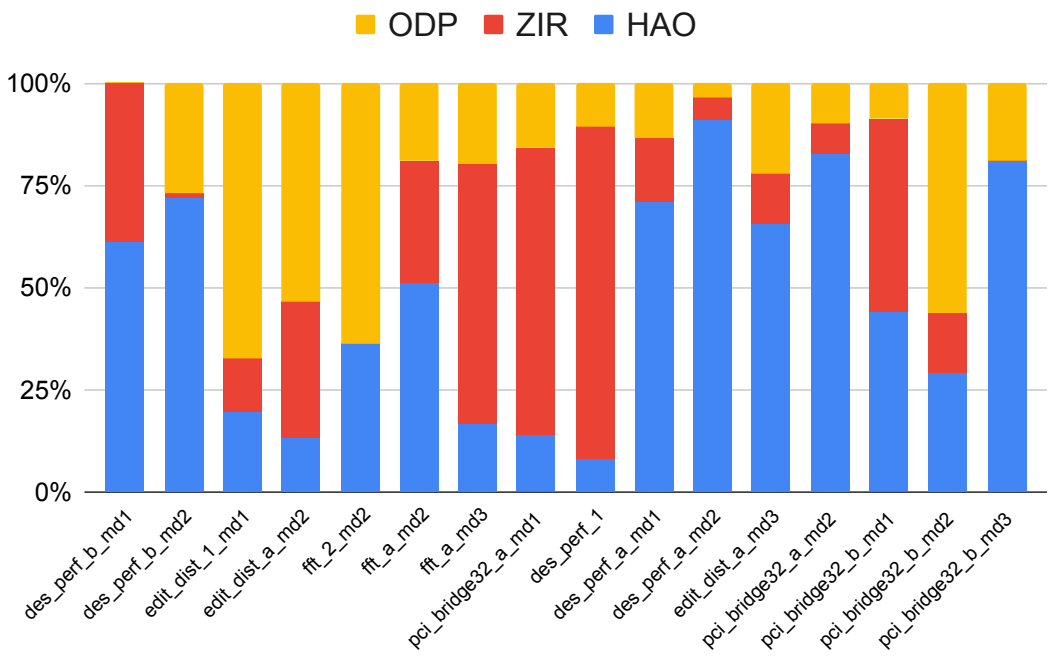

Figure 6: Distribution of best legalization algorithm in terms of wirelength variation (y axis) for each placement generated from the benchmark set (x axis).

4.3 CNN training parameters

The next step in training the CNN model is defining the learning rate to be used in order to have the smallest training loss possible. On the one hand, if the learning rate is too small, the training process may take too long. On the other hand, if the learning rate is too high, the training may not converge to a good solution. We illustrate this phenomenon in Figure 7 where we show the training loss achieved for different learning rates executed for a few iterations of the WL-CNN training. The training loss decreases as the learning rate increases until around 0.1. After this point, the training loss starts to increase, meaning that the training process is not converging anymore. As we want to choose a learning rate where the training loss is still clearly decreasing but not too close to the point where it starts diverging, we chose the learning rate of 0.01 for training both Disp-CNN and WL-CNN models.

The number of epochs is another parameter to train. As the number of epochs increases, the training loss becomes lower, but the training process takes longer and the model is more prone to over-fitting. Typically, it is beneficial to stop at the point where increasing the number of epochs does not significantly reduce the training loss, and before the validation loss starts increasing. As we divided the training process in two stages (Section 3.3), we need to choose the number of epochs for each training stage.

In Figure 8, the training and validation losses, as well as training time for 20 epochs during the first stage of the training process of WL-CNN are shown. Although the training loss keeps decreasing with the epochs, the validation loss does not show the same steady decrease. Besides the spike in the validation loss at epoch 12, we can see that the validation loss does not change

Table 3: F-scores of different CNN architectures (columns) for each class (rows) for the WL-CNN model.

	ResNet	SqueezeNet	AlexNet
HAO	0.83	0.84	0.82
ZIR	0.80	0.81	0.78
ODP	0.82	0.83	0.80
Average	0.82	0.83	0.80

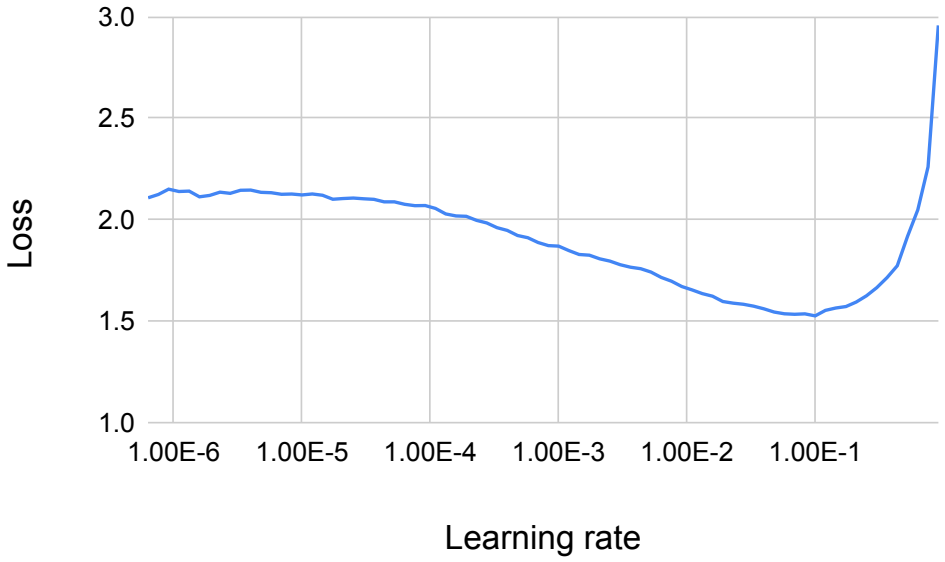

Figure 7: Training loss (x axis) for different learning rates (y axis) after a few iterations of training the WL-CNN model.

much between epochs 10 and 20. Given this situation, we chose to run the first stage of the training process for 10 epochs only, as the time it takes to train this stage for longer is not compensated by a significant loss reduction.

The second stage of the training process happens after the first stage was executed for 10 epochs and saved. As this stage is intended only for fine tuning the weights of the intermediate layers with a lower learning rate, we executed it only for 10 epochs. The measured training and validation losses, as well as training time are presented in Figure 9. We can see that the training loss slowly decreases, while the validation loss reaches its minimum value at epoch 5. This suggests that the model may be over-fitting after the fifth epoch, and that we may need more data to train this model for more epochs. As a consequence, we chose to run the second stage of training for 5 epochs only as to avoid over-fitting the network.

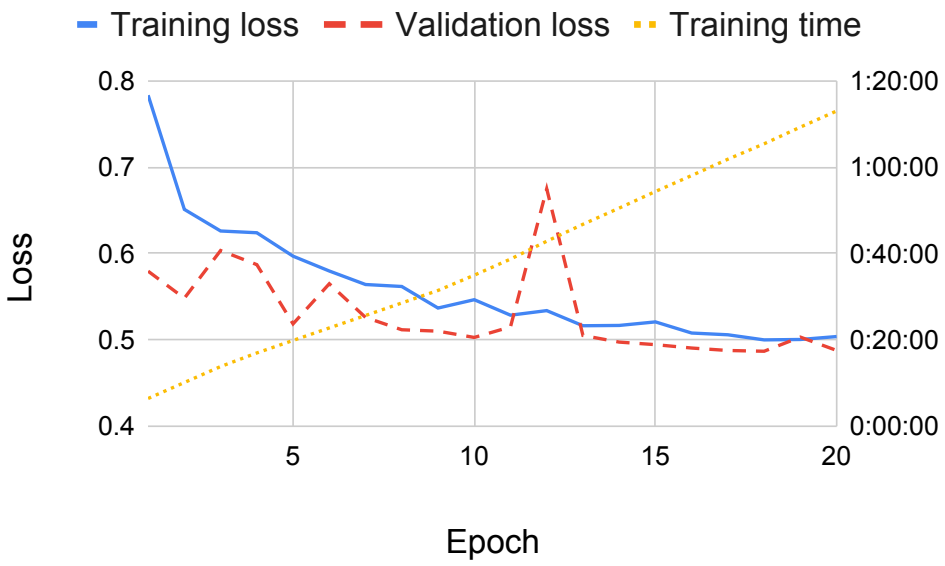

Figure 8: Training loss, validation loss and training time (y axis) at each epoch (x axis) during the first stage of the training process of WL-CNN model.

4.4 CNN image size

To investigate how much the image size impacts in the quality of the CNN models, we retrained the models with smaller images and using the same data as before. By reducing the image size, the number of sites being represented by each pixel increases, and that reduces the amount of information that the CNN model is capable of identifying. Figure 10 shows the F-score of both CNN models for 4 different image sizes: 500x500 (9 sites per pixel), 250x250 (18 sites per pixel), 150x150 (30 sites per pixel) and 50x50 (90 sites per pixel).

We observed that the Disp-CNN model is accurate until 30 sites per pixel, but degrades at 90 sites per pixel. This shows that when predicting displacement, the CNN model does not need too much information about individual sites, as it was able to correctly identify the best algorithm using only the overall layout of parts of the circuit. For the WL-CNN model, the results degrade a little at 18 sites per pixel, but they become significantly worse with 30 sites per pixel and 90 sites per pixel. This shows that the CNN model cannot correctly predict the best legalization algorithm in terms of wirelength variation with smaller images.

Those results also allow us to estimate which size of circuit we can successfully use for training given the image size the CNN supports. For example, the ICCAD 2015 CAD contest benchmark set [24] contains circuits with up to 51k sites. To use images that have at most 18 sites per pixel with those benchmarks, we need an image size of around 2800x2800 for the CNN. It is important to observe that the GPU we used in our experiments does not have enough memory to handle this image size. Therefore, to train CNN models with larger circuits, we either need a more powerful GPU or we would need to divide the circuit into smaller partitions.

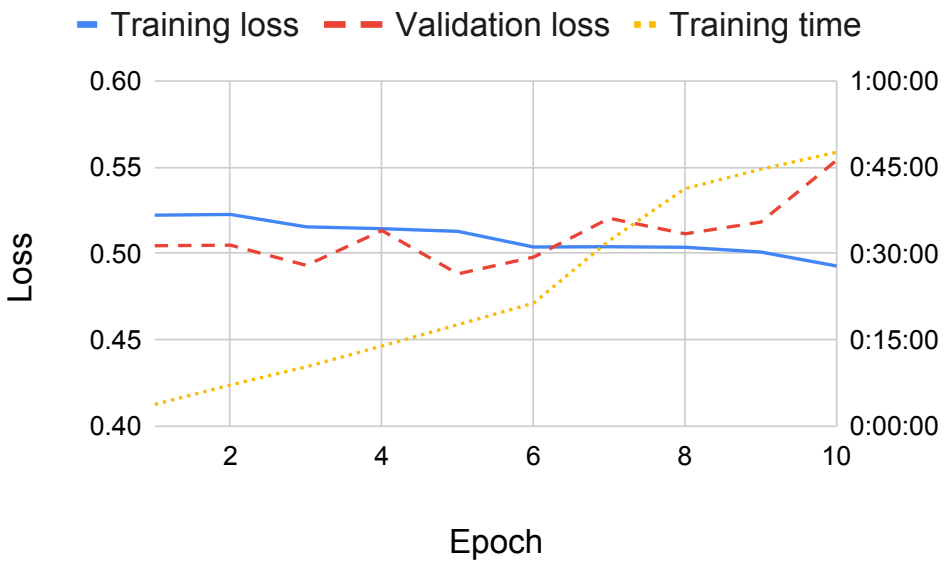

Figure 9: Training loss, validation loss and training time (y axis) at each epoch (x axis) during the second stage of the training process of WL-CNN model.

4.5 Validation of the CNN models

The trained CNN models do not have a *true* class which we are looking for. Instead, there are three classes and we want the models to identify as accurately as possible instances of each class. Hence, we analyzed the confusion matrices of both CNN models alongside their F-scores in Figure 11. In each confusion matrix, the green squares represent correctly classified instances.

We can see that the Disp-CNN model is capable of correctly classifying almost all of the instances, achieving F-scores of 0.99 for the three classes. Meanwhile, the WL-CNN model achieves lower F-scores (0.83 on average) and shows more variation among the three classes. This comes from an imbalance in the data, as about half of the instances for WL-CNN belong to the class with the highest F-score (HAO). Due to this imbalance, the model has more difficulty in classifying instances of the other classes correctly. These results expose the ability of transfer learning to properly train deep models with limited data and computing resources. Even in the case of WL-CNN where data is more imbalanced, it achieved high F-scores with just a few epochs of training. Another positive aspect is that we are able to successfully train a deep CNN model with only 13k instances (Section 4.1), with many of the CNN weights remaining stable due to the transfer learning process. If we had not used pre-trained weights, we would need to train the network for more epochs to achieve low training and validation losses. This would require longer training time, and more data to avoid over-fitting the network. By using transfer learning, the training time was less than 50 minutes, allowing us to experiment with different parameters to improve the results. This training time was the same for both Disp-CNN and WL-CNN as both used the same training methodology.

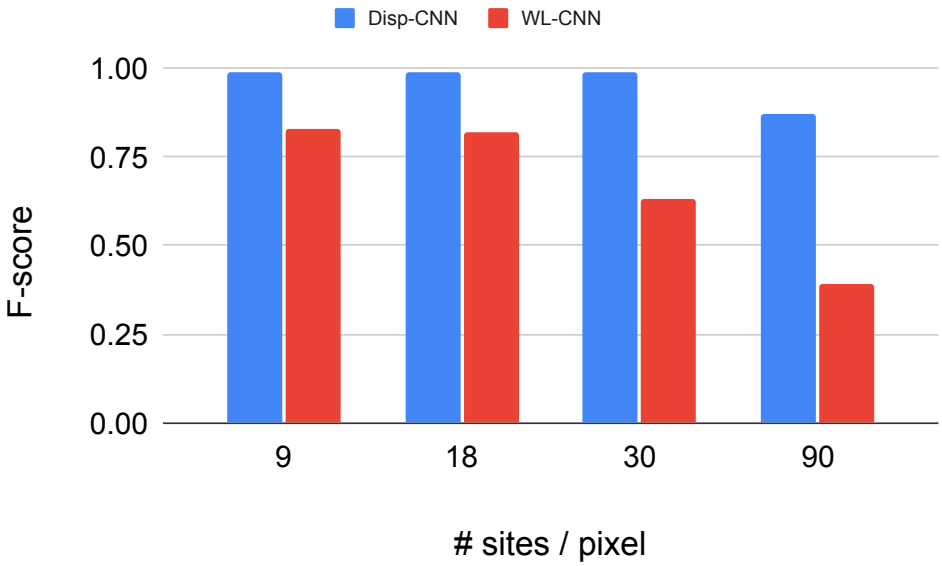

Figure 10: F-score (y axis) of the CNN models for different image sizes (x axis).

4.6 Comparison with a related non-convolutional model

This section presents a comparison with a related non-convolutional model from the state of the art [28]. This comparison is performed to ensure that our claim that CNNs provide better results than non-convolutional models is correct. In order to perform this comparison, we trained artificial neural networks (ANNs) for legalization outcome prediction using the same features from our previous work [28]. For fairness, we used as input data the same placements generated for the CNN results and the same training and validation sets, but used the feature extraction strategy from [28] instead of using the placement images. In addition, we trained the ANNs for 15 epochs (the same total number of epochs as the CNNs). We trained models for both cell displacement and wirelength variation, and we denote them as Disp-ANN and WL-ANN, respectively.

The confusion matrices and F-scores of both ANN models are presented in Figure 12.

When comparing them to the equivalent results of the Disp-CNN and WL-CNN (Figure 11 in Section 4.5), we can observe that the F-scores of the ANN models are much lower than their CNN counterparts. For Disp-ANN the F-score ranged from 0.65 to 0.82. Meanwhile, Disp-CNN achieved F-scores of at least 0.99 for all classes of the same problem. When analyzing WL-ANN, we can see that it had more difficulty than WL-CNN to handle the data imbalance of this problem. It achieved an F-score of 0.67 for HAO, and only 0.52 for ODP and 0.47 for ZIR. In comparison, WL-CNN lowest F-score was 0.81 for ZIR, which is far superior to what we see for the ANN models. Furthermore, even the least accurate of the CNN architectures tested in Section 4.2, ResNet, showed better F-scores than the ANN models (see Table 3). Finally, ANN models require more feature engineering in order to select an appropriate set of features that represent the problem, making it harder to achieve accurate results with them. Therefore, we can conclude that the CNN models are more capable of extracting important patterns from the data for these problems, and that they provide better results than non-convolutional models.

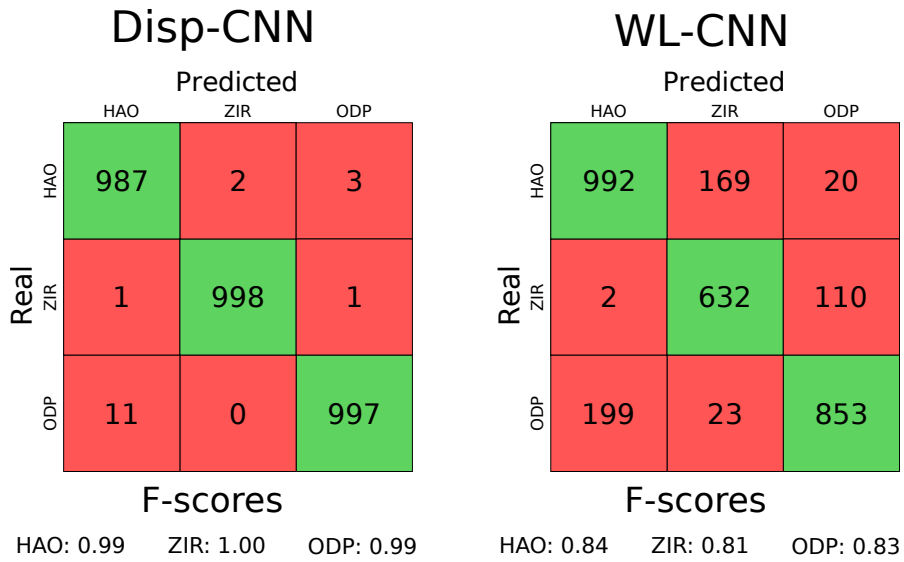

Figure 11: Confusion matrices of CNN models when cell displacement and wirelength variation as evaluation metrics. Green squares represent correct predictions, whereas red squares represent incorrect predictions.

4.7 Quality evaluation with the physical design flow

After training the Disp-CNN, WL-CNN, Disp-ANN, and WL-ANN models, we integrated them into the physical design flow as described in Figure 2. To evaluate the CNN models, we trained them using 16-fold cross validation and the same 16 thousand placements that were generated as described in Section 4.1.5. By adopting 16-fold cross validation, the models were trained using all circuits except the one being legalized, and this procedure was repeated for each of the 16 circuits. We also generated five new random placements for each circuit, so as to ensure that the models are evaluated on unseen placements. We chose five as the number of new placements to simulate a physical design flow where the legalization is applied a few times[3]. We used Algorithms 4 and 3 to implement two approaches for the algorithm selection, resulting in six legalization flow variants:

- Disp-LEG: for each placement, runs the three legalization algorithms and selects the one with the lowest cell displacement.
- Disp-CNN: for each placement, uses the Disp-CNN model to select only one legalization algorithm to execute.
- Disp-ANN: for each placement, uses a non-convolutional model to select only one legalization algorithm to execute based on cell displacement.
- WL-LEG: for each placement, runs the three legalization algorithms and selects the one with best wirelength improvement.

[3]We also evaluated how the models perform when using 10 and 20 new random placements for each circuit. The difference was of only 1% on average for the wirelength variation CNN model, and no difference was observed for the displacement model.

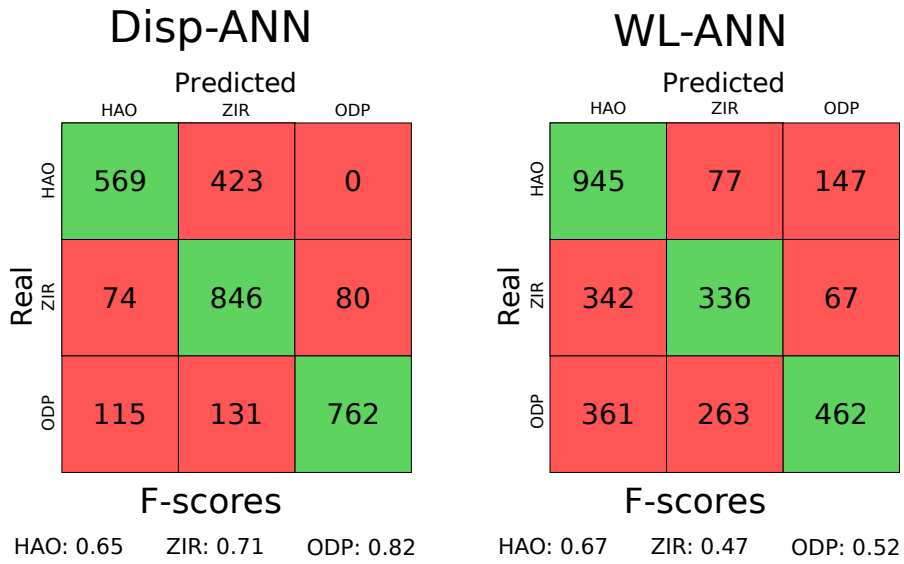

Figure 12: Confusion matrices of ANN models when using cell displacement and wirelength variation as evaluation metrics. Green squares represent correct predictions, whereas red squares represent incorrect predictions.

- WL-CNN: for each placement, uses the WL-CNN model to select only one legalization algorithm to execute.
- WL-ANN: for each placement, uses a non-convolutional model to select only one legalization algorithm to execute based on wirelength improvement.

Table 4 shows, for each circuit, the cell displacement after legalizing only with HAO, ZIR or ODP, as well as, when Disp-LEG, Disp-CNN or Disp-ANN is used to select the legalization algorithm. Since we used 16-fold cross validation, the results for each circuit were obtained from the model trained without seeing this circuit data. It is worth remarking that in this table each row brings the sum of the results of five random placements for a benchmark circuit. All the results are normalized with respect to Disp-LEG, as this encompasses the best result for each circuit. It is also important to notice that in all cases where the Disp-CNN model result is 1.00, it was because it selected the best algorithm for each of the five random placements. We did not observe cases in which the CNN model selected a slightly worse legalization algorithm, and that resulted in 1.00 due to rounding error.

The first observation is that, for each one of the three algorithms (HAO, ZIR and ODP), there is at least one case where it performs significantly worse than the best algorithm does. For instance, HAO is the best algorithm for six circuits, but also the worst for *fft_a_md*2 and *fft_a_md*3, resulting in placements that are 6% worse than Disp-LEG, on average. Similar results are seen for ZIR and ODP. This shows that there is no clear individual algorithm that is always superior to the others, which in turn emphasizes the need for a mechanism to choose the best algorithm for a given placement.

When analyzing the results of Disp-CNN, we can see that the framework achieved the same results as Disp-LEG for all circuits except *pci_bridge*32_*b_md*1. Even when choosing an incorrect legalization algorithm, the result of Disp-CNN was only 3% worse than Disp-LEG. This shows that even though the model classified some instances wrongly, it did not choose a much worse

Table 4: Results of the algorithm selection framework when using cell displacement as the evaluation metric. Each row shows the sum of cell displacement of the five random placements for each one of the benchmarks. Best results are highlighted in bold.

Circuit	Cell displacement					
	HAO	ZIR	ODP	Disp-LEG	Disp-CNN	Disp-ANN
des_perf_b_md1	**1.00**	1.05	1.19	**1.00**	**1.00**	**1.00**
des_perf_b_md2	**1.00**	1.01	1.01	**1.00**	**1.00**	**1.00**
edit_dist_1_md1	1.12	1.14	**1.00**	**1.00**	**1.00**	**1.00**
edit_dist_a_md2	1.01	**1.00**	1.04	**1.00**	**1.00**	1.01
fft_2_md2	1.08	1.23	**1.00**	**1.00**	**1.00**	**1.00**
fft_a_md2	1.24	1.22	**1.00**	**1.00**	**1.00**	1.22
fft_a_md3	1.24	1.22	**1.00**	**1.00**	**1.00**	1.22
pci_bridge32_a_md1	1.03	**1.00**	1.01	**1.00**	**1.00**	1.03
des_perf_1	1.10	**1.00**	2.65	**1.00**	**1.00**	**1.00**
des_perf_a_md1	**1.00**	1.01	1.31	**1.00**	**1.00**	**1.00**
des_perf_a_md2	**1.00**	1.02	1.21	**1.00**	**1.00**	1.02
edit_dist_a_md3	1.01	**1.00**	1.25	**1.00**	**1.00**	**1.00**
pci_bridge32_a_md2	1.03	**1.00**	1.19	**1.00**	**1.00**	1.19
pci_bridge32_b_md1	1.07	1.03	**1.00**	**1.00**	1.03	1.07
pci_bridge32_b_md2	**1.00**	**1.00**	1.01	**1.00**	**1.00**	**1.00**
pci_bridge32_b_md3	**1.00**	1.01	1.02	**1.00**	**1.00**	1.01
Average	1.06	1.06	1.18	**1.00**	**1.00**	1.05
Median	1.02	1.01	1.02	**1.00**	**1.00**	1.01

legalization algorithm for the placement. These results indicate that the framework can accurately identify the correct legalization algorithm to use, which corroborates the F-score reported in Section 4.5.

When comparing the results of Disp-CNN to Disp-ANN, we can see that using a non-convolutional model leads to large degradation in a few circuits. The most notable examples are *fft_a_md2*, *fft_a_md3* and *pci_bridge32_a_md2*, where the cell displacement was approximately 20% worse than the best result. It is also important to observe that those bad results come from circuits used in the training set. So, the Disp-ANN was not able to generalize the prediction to different placements of the same circuit.

Table 5 shows the results of evaluation with the physical design flow when using the wirelength variation as evaluation metric[4]. For each circuit, the results are normalized with respect to WL-LEG, as it always provides the best result. As for Disp-CNN (Table 4), the results for each circuit were obtained from the model trained without seeing the circuit data, and in all cases where the WL-CNN model result is 1.00, it was because the model selected the best algorithm for each of the five random placements. We can notice in the table that there are some circuits where none of the direct algorithms (HAO, ZIR, and ODP) achieves a score of 1.00 (e.g., *des_perf_b_md1* and *des_perf_a_md1*). This comes from the fact that each score is computed over five new random

[4]The wirelength variation was measured as the difference between the Steiner tree wirelength (STWL) after and before the legalization. Thus, a positive value means that the wirelength has increased.

placements for each circuit, and the best legalization algorithm can change from one random placement to the other.

Table 5: Results of the algorithm selection framework when using wirelength variation as the evaluation metric. Each row shows the sum of wirelength variation of the five random placements for each one of the benchmarks. Best results are highlighted in bold.

Circuit	Wirelength variation					
	HAO	ZIR	ODP	WL-LEG	WL-CNN	WL-ANN
des_perf_b_md1	1.03	1.04	1.77	**1.00**	1.03	1.03
des_perf_b_md2	**1.00**	1.23	1.03	**1.00**	**1.00**	**1.00**
edit_dist_1_md1	1.08	1.11	**1.00**	**1.00**	**1.00**	1.08
edit_dist_a_md2	1.03	1.01	1.02	**1.00**	1.02	1.03
fft_2_md2	1.01	1.10	**1.00**	**1.00**	**1.00**	1.01
fft_a_md2	1.02	1.06	1.07	**1.00**	1.03	1.02
fft_a_md3	1.07	1.03	1.07	**1.00**	1.03	1.07
pci_bridge32_a_md1	1.04	**1.00**	1.04	**1.00**	**1.00**	1.04
des_perf_1	1.02	**1.00**	3.99	**1.00**	**1.00**	**1.00**
des_perf_a_md1	1.04	1.15	6.04	**1.00**	1.04	1.04
des_perf_a_md2	**1.00**	1.75	6.84	**1.00**	**1.00**	1.75
edit_dist_a_md3	**1.00**	1.12	1.37	**1.00**	**1.00**	1.12
pci_bridge32_a_md2	**1.00**	1.20	2.10	**1.00**	**1.00**	2.10
pci_bridge32_b_md1	1.06	1.01	1.08	**1.00**	1.06	1.06
pci_bridge32_b_md2	1.02	1.02	**1.00**	**1.00**	**1.00**	1.02
pci_bridge32_b_md3	**1.00**	1.05	1.03	**1.00**	**1.00**	**1.00**
Average	1.03	1.12	2.03	**1.00**	1.01	1.15
Median	1.02	1.05	1.07	**1.00**	**1.00**	1.03

We can observe in Table 5 that HAO achieves better average and median scores for the wirelength metric, which was not the case for cell displacement. However, if we were to use only HAO to legalize all the circuits, we would still lose on legalization quality for some circuits (such as $edit\_dist\_1\_md1$ and fft_a_md3), as HAO delivers the best solution for only 5 out of 16 circuits.

Although ZIR and ODP performed the best for some circuits, there were also cases where they performed very poorly. For example, ODP is more than 6 times worse than WL-LEG for $des\_perf\_a\_md1$ and $des\_perf\_a\_md2$. When using the framework to predict which algorithm to use, it is especially important for the model to be able to avoid these extreme situations so as not to compromise the solution quality.

Shifting our focus to the results with WL-CNN, we can see that it chose the best legalization algorithms for 10 out of the 16 circuits, and it performed as well as any individual direct algorithm in 13 of these cases. Among the other 3 benchmarks, the results for $edit\_dist\_a\_md2$ are very similar among the three legalization algorithms, which makes it harder for the model to identify the best option during execution. For fft_a_md2, we observe that the model chose correctly for few of the random placements, but chose incorrectly for others, which left it with a score

different to the other algorithms. Finally, *pci_bridge*32_*b_md*1 is the same circuit that Disp-CNN classified incorrectly. This indicates that the CNN model is having some difficulty extracting the necessary features to predict for this circuit. Nevertheless, WL-CNN provided results that were better than any individual legalization algorithm with an average result that was only 1% worse than WL-LEG.

When comparing WL-CNN to the WL-ANN model, the non-convolutional model achieved the best result only in 3 out of 16 circuits, and it was significantly worse in a few circuits. For example, for *des_perf_a_md*2 it resulted in 75% of wirelength degradation whereas for *pci_bridge*32_*a_md*2 the wirelength was more than two times worse than the best. As a result, the WL-ANN option was worse than just running HAO for all circuits.

All these results demonstrate that the CNN models are accurate and effective predictors, being much better than the non-convolutional model. They also show that the proposed algorithm selection framework is robust and can predict the best algorithm not only for cell displacement metric, but also for minimizing wirelength variation. This is a great advantage of using CNNs for prediction, as we did not need to perform specific feature engineering to handle different metrics. By using the same input features (placement images), the CNN was able to extract the patterns that are important for each metric, and to achieve an accuracy close to the optimal one.

4.8 Speedup of the proposed CNN models

The main advantage of using our framework is not having to run all legalization algorithms every time to choose the best one. This is important because physical design is usually constrained by the execution time of its underlying algorithms, which can make running multiple algorithms for a single step infeasible. The difference between using the CNN model and running all legalization algorithms is depicted in Figure 13 and Tables 6 and 7. The figure illustrates the speedup as the average execution time ratio of LEG and CNN algorithms for each metric (cell displacement and wirelength variation) and each circuit. A speedup greater than 1 means that the CNN-based approach is faster than the baseline (LEG selector). The tables present the runtime in seconds of each algorithm and each algorithm selector for cell displacement and wirelength variation, respectively. The runtimes and speedups were measured as the mean of 10 executions for each one of the five random placements of each circuit.

The variation in speedup among the circuits is very noticeable in Figure 13, with some circuits having a speedup close to 2x, while the speedup of WL-CNN for *edit_dist_a_md*3 is over 10. This variation comes from the different execution time of each possible legalization algorithm. In some situations, the best algorithm to use is also the slowest, which constrains the performance gain. But in other situations, the best algorithm is also the fastest. For example, ODP is faster than the other two algorithms for almost all circuits. For those cases, whenever ODP is chosen because it is the best algorithm, the speedup will be higher as the algorithm selection framework does not need to run the slower algorithms. However, if ODP is not the best algorithm, running only ODP would be faster than running the algorithm selection framework, but this will result in worse legalization quality.

This also explains the difference in speedup between Disp-CNN and WL-CNN for some circuits, as the models can choose different algorithms for the different metrics. A notable case is *edit_dist_a_md*3, for which the best algorithm for cell displacement was mostly ZIR, that is more than ten times slower than the other two algorithms for this circuit, resulting in a small speedup for Disp-CNN. On the other hand, the best algorithm for wirelength in this circuit was mostly HAO, resulting in the large speedup of WL-CNN.

We also observed that the execution time of the CNN model's inference itself is very small compared to the execution time of the legalization algorithms. It takes only a few seconds

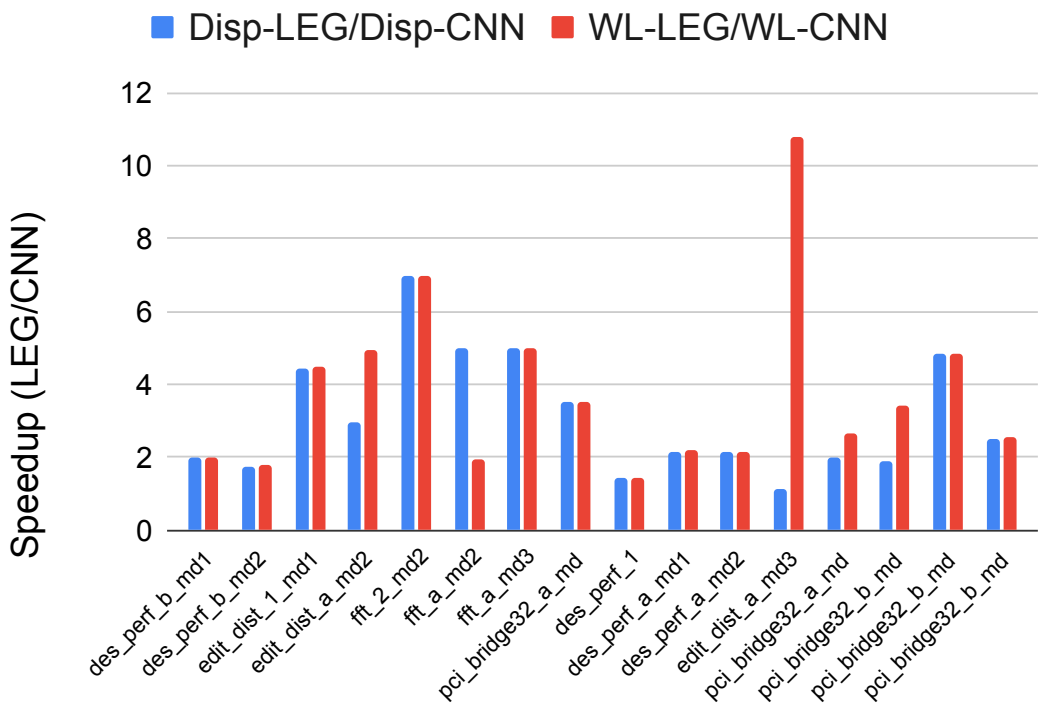

Figure 13: Speedup when using the algorithm selection framework (y axis) for each circuit (x axis). Results for the cell displacement model are shown in blue, and results for the wirelength model are shown in red.

to run the CNN model inference, while it might take a few minutes to run the legalization model. The inference runtime ranges from 25% of the total runtime when the best algorithm is fast for the circuit (such as *edit_dist_1_md1*), to less than 1% when the best algorithm is slow (such as *edit_dist_a_md3*). As the inference process is faster than even the fastest legalization algorithm, the framework always provides a better performance than running all three legalization algorithms. It is important to observe that the speedup achieved by using the framework scales with the number of legalization algorithms. For example, if we were to double the number of legalization algorithms considered (and the new algorithms had roughly the same total execution times as the algorithms already considered), we could expect the execution time of the LEG versions of the algorithm selection to double, while the execution time of the CNN version would remain mostly constant. Therefore, the speedup would double as well.

Finally, even though we could reduce the runtime of the LEG versions by running the algorithms in parallel, this would not result in a great benefit for the circuits with large speedup. This is because for those circuits the runtime of LEG is dominated by the slowest legalization algorithm, which is much slower than the other two. Since the ML model only has to run one of the algorithms, it would still speed up the algorithm selection process.

Table 6: Runtime in seconds of each legalization algorithm and algorithm selector for cell displacement. Best results are highlighted in bold.

Circuit	Runtime (s)					
	HAO	ZIR	ODP	Disp-LEG	Disp-CNN	Disp-ANN
des_perf_b_md1	51	35	**27**	113	56	64
des_perf_b_md2	60	30	**25**	116	65	76
edit_dist_1_md1	60	32	**20**	112	25	32
edit_dist_a_md2	67	37	**20**	124	42	77
fft_2_md2	20	10	**5**	35	**5**	**5**
fft_a_md2	15	**5**	**5**	25	**5**	7
fft_a_md3	15	**5**	**5**	25	**5**	8
pci_bridge32_a_md1	15	8	**5**	28	8	20
des_perf_1	80	222	**25**	327	226	206
des_perf_a_md1	65	**40**	48	153	71	79
des_perf_a_md2	65	50	**35**	150	70	50
edit_dist_a_md3	74	773	**20**	867	762	172
pci_bridge32_a_md2	15	20	**5**	40	20	**5**
pci_bridge32_b_md1	17	31	**10**	58	31	19
pci_bridge32_b_md2	29	**10**	**10**	49	**10**	**10**
pci_bridge32_b_md3	20	17	**12**	49	19	15
Average	42	83	**17**	142	89	53
Median	40	31	**16**	85	28	26

5 Conclusions

In this work, we proposed a legalization algorithm selection framework using CNN. We proposed both feature extraction and training methodology that adapt the training of deep CNNs to a physical design problem. We provided details of the training process and choice of parameters so that the proposed framework can be used by other researchers in applications that rely on placement data. We evaluated the proposed framework by training a model to decide between three state-of-the-art legalization algorithms for different circuit placements. The CNN model was integrated in a physical design flow to evaluate how such predictions can improve the legalization quality according to cell displacement and wirelength variation. The proposed CNN model achieved an F-score of 0.98 when predicting cell displacement and at least 0.70 when predicting wirelength variation. The proposed algorithm selection framework achieved the best results in 15 out of 16 designs when evaluating cell displacement and 10 out of 16 when evaluating wirelength variation, being better than running each one of the legalization algorithms separately and being better than a related non-convolutional model. In addition, using the framework significantly reduced the execution time up to 10x compared to running all three algorithms to select the best one.

As future work, regression models can be used to predict the actual results of the algorithms being compared. This would allow the CNN model to be used in more physical design applications. The proposed framework can also be extended to perform algorithm selection according

Table 7: Runtime in seconds of each legalization algorithm and algorithm selector for wirelength variation. Best results are highlighted in bold.

Circuit	Runtime (s)					
	HAO	ZIR	ODP	Disp-LEG	Disp-CNN	Disp-ANN
des_perf_b_md1	52	35	**27**	114	57	60
des_perf_b_md2	60	31	**25**	116	66	70
edit_dist_1_md1	60	32	**20**	112	25	70
edit_dist_a_md2	67	37	**20**	124	25	77
fft_2_md2	20	10	**5**	35	**5**	21
fft_a_md2	15	**5**	**5**	25	13	15
fft_a_md3	15	**5**	**5**	25	**5**	17
pci_bridge32_a_md1	15	8	**5**	28	8	19
des_perf_1	80	251	**25**	356	251	272
des_perf_a_md1	66	**40**	48	154	71	100
des_perf_a_md2	65	50	**35**	150	70	55
edit_dist_a_md3	74	762	**20**	856	79	239
pci_bridge32_a_md2	15	20	**5**	40	15	6
pci_bridge32_b_md1	16	31	**10**	58	17	30
pci_bridge32_b_md2	29	**10**	**10**	49	**10**	43
pci_bridge32_b_md3	19	18	**12**	49	19	32
Average	42	84	**17**	143	46	70
Median	40	31	**16**	85	22	49

to other metrics, or in the context of other physical design steps, such as global routing.

6 ACKNOWLEDGMENTS

This study was financed by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001 and Capes-PrInt Program (grant number 88881.310783/2018-01), by the Brazilian Council for Scientific and Technological Development (CNPq) PQ grant 312077/2018-1, and by the Natural Science and Engineering Council of Canada grant number 10015685. We also would like to thank Haocheng Li, Ziran Zhu and the OpenROAD team for providing their legalization algorithms. This work was partially developed while Laércio L. Pilla was a member of the *Laboratoire de Recherche en Informatique*.

References

[1] Opendp: Open source detailed placement engine, https://github.com/The-OpenROAD-Project/OpenDP. [Online; accessed 12-October-2020].

[2] Ophidian: an open source library for physical design research and teaching, https://gitlab.com/renan.o.netto/ophidian-research. [Online; accessed 12-January-2021].

[3] Erick Carvajal Barboza, Nishchal Shukla, Yiran Chen, and Jiang Hu. Machine learning-based pre-routing timing prediction with reduced pessimism. In *DAC*, pages 1–6. IEEE, 2019.

[4] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[5] Wei-Ting J Chan, Pei-Hsin Ho, Andrew B Kahng, and Prashant Saxena. Routability optimization for industrial designs at sub-14nm process nodes using machine learning. In *ISPD*, pages 15–21. ACM, 2017.

[6] Jianli Chen, Ziran Zhu, Wenxing Zhu, and Yao-Wen Chang. Toward optimal legalization for mixed-cell-height circuit designs. In *DAC*, pages 1–6. ACM, 2017.

[7] Wing-Kai Chow, Chak-Wa Pui, and Evangeline FY Young. Legalization algorithm for multiple-row height standard cell design. In *DAC*, pages 1–6. IEEE, 2016.

[8] Nima Karimpour Darav, I Bustany, Andrew Kennings, and Ravi Mamidi. Iccad-2017 cad contest in multi-deck standard cell legalization and benchmarks. In *ICCAD*, 2017.

[9] SangGi Do, Mingyu Woo, and Seokhyeong Kang. Fence-region-aware mixed-height standard cell legalization. In *GLSVLSI*, pages 259–262, 2019.

[10] Brian S Everitt. *The Cambridge dictionary of statistics*, volume 106. Cambridge University Press, 2006.

[11] Upma Gandhi, Ismail Bustany, William Swartz, and Laleh Behjat. A reinforcement learning-based framework for solving physical design routing problem in the absence of large test sets. In *MLCAD*, pages 1–6. IEEE, 2019.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016. http://www.deeplearningbook.org.

[13] Haipeng Guo and William H Hsu. *Algorithm selection for sorting and probabilistic inference: a machine learning-based approach*. PhD thesis, Kansas State University, 2003.

[14] Seung-Soo Han, Andrew B Kahng, Siddhartha Nath, and Ashok S Vydyanathan. A deep learning methodology to proliferate golden signoff timing. In *DATE*, pages 1–6. IEEE, 2014.

[15] Jeremy Howard and Sebastian Ruder. Fine-tuned language models for text classification. *CoRR*, pages 1–12, 2018.

[16] Jeremy Howard and Rachel Thomas. fast.ai - Making neural networks uncool again. http://www.fast.ai/, 2018. [Online; accessed 12-October-2020].

[17] Chung-Yao Hung, Peng-Yi Chou, and Wai-Kei Mak. Mixed-cell-height standard cell placement legalization. In *GLSVLSI*, pages 149–154. ACM, 2017.

[18] Wei-Tse Hung, Jun-Yang Huang, Yih-Chih Chou, Cheng-Hong Tsai, and Mango Chao. Transforming global routing report into drc violation map with convolutional neural network. In *ISPD*, pages 57–64. ACM, 2020.

[19] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint*, pages 1–13, 2016.

[20] Andrew B Kahng, Bill Lin, and Siddhartha Nath. High-dimensional metamodeling for prediction of clock tree synthesis outcomes. In *SLIP*, pages 1–7. IEEE, 2013.

[21] Andrew B Kahng, Mulong Luo, and Siddhartha Nath. Si for free: machine learning of interconnect coupling delay and transition effects. In *SLIP*, pages 1–8. IEEE, 2015.

[22] Andrew B Kahng, Uday Mallappa, and Lawrence Saul. Using machine learning to predict path-based slack from graph-based timing analysis. In *ICCD*, pages 603–612. IEEE, 2018.

[23] Pascal Kerschke and Heike Trautmann. Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning. *Evolutionary computation*, 27(1):99–127, 2019.

[24] M. Kim, J. Hu, J. Li, and N. Viswanathan. ICCAD-2015 CAD contest in incremental timing-driven placement and benchmark suite. 2015.

[25] Lars Kotthoff, Ian P Gent, and Ian Miguel. An evaluation of machine learning in algorithm selection for search problems. *AI Communications*, 25(3):257–270, 2012.

[26] Haocheng Li, Wing-Kai Chow, Gengjie Chen, Evangeline FY Young, and Bei Yu. Routability-driven and fence-aware legalization for mixed-cell-height circuits. In *DAC*, pages 1–6. ACM, 2018.

[27] Rongjian Liang, Hua Xiang, Diwesh Pandey, Lakshmi Reddy, Shyam Ramji, Gi-Joon Nam, and Jiang Hu. Drc hotspot prediction at sub-10nm process nodes using customized convolutional network. In *ISPD*, pages 135–142. ACM, 2020.

[28] Renan Netto, Sheiny Fabre, Tiago Fontana, Vinicius Livramento, Laércio Lima Pilla, and José Güntzel. How deep learning can drive physical synthesis towards more predictable legalization. In *ISPD*, pages 3–10, 2019.

[29] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[30] Yutaka Sasaki. The truth of the f-measure. *Teach Tutor mater*, 1(5):1–5, 2007.

[31] Kate A Smith-Miles. Cross-disciplinary perspectives on meta-learning for algorithm selection. *CSUR*, 41(1):1–25, 2009.

[32] Aysa Fakheri Tabrizi, Logan Rakai, Nima Karimpour Darav, Ismail Bustany, Laleh Behjat, Shuchang Xu, and Andrew Kennings. A machine learning framework to identify detailed routing short violations from a placed netlist. In *DAC*, pages 1–6. IEEE, 2018.

[33] Chao-Hung Wang, Yen-Yi Wu, Jianli Chen, Yao-Wen Chang, Sy-Yen Kuo, Wenxing Zhu, and Genghua Fan. An effective legalization algorithm for mixed-cell-height standard cells. In *ASP-DAC*, pages 450–455. IEEE, 2017.

[34] Zhiyao Xie, Yu-Hung Huang, Guan-Qi Fang, Haoxing Ren, Shao-Yun Fang, Yiran Chen, and Jiang Hu. Routenet: routability prediction for mixed-size designs using convolutional neural network. In *ICCAD*, pages 1–8. IEEE, 2018.

[35] Tao-Chun Yu, Shao-Yun Fang, Hsien-Shih Chiu, Kai-Shun Hu, Philip Hui-Yuh Tai, Cindy Chin-Fang Shen, and Henry Sheng. Pin accessibility prediction and optimization with deep learning-based pin pattern recognition. In *DAC*, pages 1–6. IEEE, 2019.

[36] Tao-Chun Yu, Shao-Yun Fang, Hsien-Shih Chiu, Kai-Shun Hu, Philip Hui-Yuh Tai, Cindy Chin-Fang Shen, and Henry Sheng. Lookahead placement optimization with cell library-based pin accessibility prediction via active learning. In *ISPD*, pages 65–72. ACM, 2020.

[37] Quan Zhou, Xueyan Wang, Zhongdong Qi, Zhuwei Chen, Qiang Zhou, and Yici Cai. An accurate detailed routing routability prediction model in placement. In *ASQED*, pages 119–122. IEEE, 2015.

[38] Zhonghua Zhou, Ziran Zhu, Jianli Chen, Yuzhe Ma, Bei Yu, Tsung-Yi Ho, Guy Lemieux, and Andre Ivanov. Congestion-aware global routing using deep convolutional generative adversarial networks. In *MLCAD*, pages 1–6. IEEE, 2019.

[39] Ziran Zhu, Xingquan Li, Yuhang Chen, Jianli Chen, Wenxing Zhu, and Yao-Wen Chang. Mixed-cell-height legalization considering technology and region constraints. In *ICCAD*, pages 1–8. IEEE, 2018.