UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Renan Oliveira Netto

Algorithm Selection Framework for Legalization Using Deep Convolutional Neural Networks

Florianópolis
2022

Renan Oliveira Netto

Algorithm Selection Framework for Legalization Using Deep Convolutional Neural Networks

Tese submetida ao Programa de Pós-Graduação em Ciência da Computação para a obtenção do título de Doutor em Ciência da Computação.
Orientador: Prof. José Luís Almada Güntzel, Dr.
Coorientadores: Vinicius Livramento, Dr. and Laleh Behjat, Dr.

Florianópolis
2022

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Netto, Renan
Algorithm Selection Framework for Legalization Using Deep Convolutional Neural Networks / Renan Netto ; orientador, José Luís Almada Güntzel, coorientador, Vinicius Livramento, 2022.
81 p.

Tese (doutorado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Ciência da Computação, Florianópolis, 2022.

Inclui referências.

1. Ciência da Computação. 2. Electronic Design Automation. 3. Machine Learning. 4. Physical Design. 5. Legalization. I. Almada Güntzel, José Luís. II. Livramento, Vinicius. III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. IV. Título.

Renan Oliveira Netto

Algorithm Selection Framework for Legalization Using Deep Convolutional Neural Networks

Florianópolis, 2022.

ACKNOWLEDGEMENTS

I would like to acknowledge my parents Carlos Alberto Barbosa Netto and Loreni Oliveira Netto, as well as my brother Caio Oliveira Netto for supporting me on this journey through the University since 2010. I would also like to acknowledge my laboratory colleagues, specially Tiago Augusto Fontana and Sheiny Fabre Almeida for all the help during my research. Third, I would like to acknowledge my advisors for their suggestions to complete this thesis. Finally, this work was done with the support from Fundação de Amparo à Pesquisa e Inovação do Estado de Santa Catarina (FAPESC).

RESUMO

Modelos de *machine learning* tem sido usados para melhorar a qualidade de diferentes etapas da síntese física de circuitos integrados, tais como análise de *timing*, síntese da árvore de relógio e roteamento. No entanto, poucos trabalhos tem abordado o problema de seleção de algoritmos durante a síntese física, o qual pode reduzir drasticamente o esforço computacional de algumas etapas. Este trabalho propõe um framework para seleção de algoritmos usando redes neurais convolucionais profundas. Para extrair features, foram utilizadas imagens de posicionamentos de circuitos simulando três etapas diferentes do fluxo de síntese física. Após, os modelos foram treinados aplicando-se aprendizado por transferência, usando pesos pré-treinados da arquitetura Squeezenet. O uso de aprendizado por transferência reduz significativamente o tempo de treinamento e a quantidade de dados necessários, mesmo quando a rede tenha sido pré-treinada para uma aplicação diferente. Uma extensa análise experimental de modelos de *machine learning* foi realizada, provendo detalhes sobre o de como os parâmetros do modelo foram escolhidos, incluindo a definição da arquitetura da rede neural convolucional, o tamanho das imagens, o fator de aprendizado e o número de épocas. O framework proposto foi avaliado treinando-se um modelo para selecionar entre diferentes algoritmos de legalização de acordo com duas métricas, deslocamento de células e variação do comprimento de interconexões, e considerando três cenários distintos de movimentação de células. Os modelos treinados atingiram F-score entre 0,90 e 1,00 para deslocamento de células e entre 0,81 e 0,97 para comprimento de interconexões, demonstrando precisão mesmo na presença de dados desbalanceados. Os modelos treinados também foram superiores a um modelo não convolucional para ambas métricas consideradas. Os modelos convolucionais treinados foram utilizados para construir uma técnica para seleção de algoritmo de legalização. Quando integrado a um fluxo de síntese física, o seletor de algoritmo proposto atingiu o melhor resultado na maioria dos circuitos testados, sendo melhor que qualquer algoritmo, individualmente. Mesmo quando o seletor proposto não atinge o melhor resultado, seu resultado ainda é apenas 7% pior, no máximo, mostrando que ele pode ser utilizado sem comprometer a qualidade do fluxo de síntese física. Finalmente, observa-se que o uso do modelo de *machine learning* proposto acelerou o processo de seleção de algoritmos em até 10x quando comparado a executar os algoritmos separadamente.

Palavras-chave: *Electronic Design Automation* (EDA). Seleção de algoritmos. *Machine Learning* (ML). Redes neurais convolucionais. Síntese física. Legalização *multi-row*.

RESUMO EXPANDIDO

Introdução

Os avanços no processo de fabricação de circuitos integrados VLSI (*Very Large-Scale Integration*) permitiram o aumento do número de transistores por área de silício em um chip. Esse aumento é conhecido como a lei de Moore, e prevê que o número de transistores que podem ser integrados em uma pastilha de silício (*chip*) dobra aproximademente a cada dois anos. Porém, este aumento do número de transistores por *chip* traz consigo novos desafios para o projeto de circuitos integrados, os quais vêm dificultando de maneira progressiva o pleno aproveitamente desta capacidade de integração.
Um dos principais fatores limitantes para o projeto de circuitos integrados mais complexos é a imprevisibilidade do fluxo de projeto, no que se refere às diversas figuras de mérito de projeto (ou métricas), tais como potência, desempenho e área (normalmente referenciadas pela sigla PPA). Tal imprevisibilidade pode exigir que o projetista precise iterar diversas vezes durante as etapas do fluxo de projeto para satisfazer os objetivos de PPA. Isto pode diminuir a qualidade final do circuito, pois frequentemente os projetistas precisam fazer estimativas pessimista ao longo das etapas de projeto para evitar comprometer alguma etapa futura. Portanto, é importante aumentar a previsibilidade do fluxo de projeto.
Uma das formas de aumentar esta previsibilidade é através do uso de aprendizado de máquina (em inglês, *machine learning* - ML). ML já tem sido usado por diversos trabalhos na área de síntese física com este objetivo, porém poucos trabalhos a aplicaram para selecionar o algoritmo a ser utilizado em alguma das diferentes etapas do fluxo. Uma das etapas onde isto pode ser feito é na etapa de legalização. A complexidade de circuitos VLSI modernos traz consigo novos desafios durante a legalização. Portanto, poder escolher entre diversos algoritmos permite melhorar a qualidade final do circuito.

Objetivos

Recentemente, redes neurais convolucionais (em inglês, *Convolutional Neural Networks* - CNNs) se tornaram populares especialmente para análise de dados com imagens. Isto se deve ao fato destas redes serem capazes de extrair padrões espaciais dos dados. Como o posicionamento de um circuito integrado é um conjunto de dados espaciais, CNNs podem ser utilizadas para realizarem previsões através destes dados.
Portanto, o objetivo deste trabalho de doutorado é utilizar CNNs para realizar a seleção de algoritmos de legalização durante o projeto de um circuito integrado. Espera-se que o uso da CNN permita fazer esta seleção de forma mais rápida do que executando diversos algoritmos de legalização.

Metodologia

Para tanto, este trabalho de doutorado seguiu os seguintes passos metodológicos: 1) extraição de *features* de circuitos em forma de imagens; 2) seleção dos parâmetros de treinamento da CNN através de experimentação; 3) treinamento e validação das CNNs; 4) integração das CNNs no fluxo de síntese física; 5) validação da hipótese de pesquisa através de experimentação do fluxo com seleção de algoritmos.
Foram utilizados três algoritmos do estado da arte de legalização, dado que cada um deles apresenta melhor desempenho em casos diferentes. A ferramenta utilizada para treinar as CNNs é a biblioteca *fast.ai*, e os circuitos utilizados nos experimentos são o conjunto de *benchmarks* da competição de legalização de circuitos com células de múltiplos níveis

realizado pela conferência *ICCAD*.
Para extrair uma grande quantidade de dados a partir de uma quantidade limitada de circuitos disponíveis, foram aplicados vetores de movimentos aleatórios aos circuitos. Além disso, foram utilizadas três estratégias de movimento diferentes para simular diferentes etapas do fluxo de projeto físico. Desta forma, a hipótese de pesquisa pode ser verificada em diferentes condições.

Resultados e Discussão
A primeira sequência de experimentos realizados foi para selecionar os parâmetros de treinamento das CNNs. Os parâmetros selecionados foram: arquitetura da rede, taxa de aprendizado, número de épocas de treinamento e tamanho da imagem de entrada. Esta metodologia ainda pode ser utilizada como base para outros trabalhos que utilizem CNNs no contexto de síntese física. Em especial, os experimentos com o tamanho de imagens permite estimar que tamanho de imagem seria necessário para circuitos maiores.
Após selecionar os parâmetros de treinamento, foi feita uma extensa análise experimental simulando três diferentes cenários de movimentação de células: 1) movimentação simulando um posicionamento global (*cell movement simulating global placement* - CMGP); 2) movimentação simulando um posicionamento detalhado (*ell movement simulating detailed placement* - CMDP); 3) movimentação simulando posicionamento incremental guiado por atraso (*cell movement simulating incremental timing-driven placement* - CMITDP). Para validar as CNNs de cada cenário foram medido seus F-scores para os conjuntos de validação. As CNNs atingiram F-scores de pelo menos 0,81 para CMGP, pelo menos 0,83 para CMDP e pelo menos 0,81 para CMITDP, chegando próximo de 1,00 em alguns casos. Estes resultados mostram que as redes são capazes de prever o algoritmo correto a ser utilizado mesmo na presença de dados desbalanceados.
Particularmente para o cenário CMGP, os resultados foram comparados com uma rede neural não convolucional (*Artificial Neural Network* - ANN). A ANN atingiu resultados significativamente inferiores, com F-scores de até 0,47, ao passo que a CNN atingiu pelo menos 0,81. Estes resultados confirmam a hipótese de que CNNs são mais adequadas para este tipo de dados, descartando a necessidade de mais experimentação com ANNs.
O último conjunto de experimentos teve como objetivo avaliar a qualidade das previsões usando CNNs quando estas estão integradas no fluxo de síntese física. Para a maioria dos circuitos testados, a seleção de algoritmos utilizando CNN atingiu exatamente os mesmos resultados que a seleção baseada na execução dos três algoritmos de legalização. Para os circuitos onde isso não ocorreu, a degradação do uso das CNNs foi de, no máximo, tão somente 7%, indicando que as CNNs podem ser utilizadas como ferramenta de seleção de algoritmos sem comprometer a qualidade do fluxo de projeto.
Por fim, foi observado que utilizar as CNNs para realizar a seleção de algoritmos de legalização foi sempre mais rápido do que executar os três algoritmos de legalização disponíveis. Por exemplo, o tempo médio de execução do seletor de algoritmos utilizando CNNs foi de 28 a 46 segundos, ao passo que, para a versão sem CNN foi de 90 a 143 segundos. Para alguns circuitos, utilizar a CNN ofereceu uma aceleração de até 10 vezes do tempo de execução.

Considerações Finais
Esta tese de doutorado apresentou uma estratégia de seleção de algoritmos de legalização utilizando CNNs para prever qual algoritmo a ser utilizado sem a necessidade de executar todos os algoritmos disponíveis. Esta estratégia se monstrou eficiente nos experimentos

realizados, permitindo a melhora da qualidade do fluxo de projeto físico de circuitos integrados.
No entanto, ainda há pontos a serem investigados em trabalhos futuros. Primeiro, a mesma estratégia de seleção de algoritmos pode ser utilizada com diferentes métricas de avaliação. Para isto, basta modificar o processo de extração das *features* para classificar os dados utilizando outra métrica. De maneira semelhante, é possível testar a estratégia de seleção naseada em CNN proposta usando outros conjuntos de circuitos de teste, através da geração de dados novos para eles, porém retreinando as CNNs.
Uma outra possível investigação reside no uso da seleção de algoritmos com legalizadores de ferramentas industriais. Algoritmos industriais são tipicamente mais poderosos, porém mais lentos do que ferramentas acadêmicas e portanto, não é possível fazer uma comparação justa entre ferramentas industriais e acadêmicas. Porém, é possível comparar algoritmos industriais de diferentes ferramentas. Desta forma, um projetista com acesso a diferentes ferramentas poderia selecionar a melhor opção para um circuito específico.
Outra possibilidade de trabalho futuro consiste na modelagem da seleção de algoritmos como um problema de regressão. Nesta tese a seleção foi modelada como um problema de classificação, onde cada algoritmo representa uma classe possível. Ao modelar como um problema de regressão é possível selecionar o algoritmo combinando diferentes métricas de acordo com as necessidades do projeto.
Por fim, um grande desafio que permanece em aberto para o uso de ML durante a síntese física é como generalizar o modelo treinado para outras tecnologias. Nesta tese, todos os circuitos testados haviam sido projetados para a mesma tecnologia de fabricação. Usar um modelo já treinado nas tecnologias de fabricação mais recentes tende a ser significativamente mais difícil, pois tais tecnologias introduzem novos desafios ao projeto. Portanto, este é um desafio não só para a presente pesquisa, mas também para outros trabalhos que utilizam ML durante a síntese física.

Palavras-chave: *Electronic Design Automation* (EDA). Seleção de algoritmos. *Machine Learning* (ML). Redes neurais convolucionais. Síntese física. Legalização *multi-row*.

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, very few works have addressed the problem of algorithm selection during physical design. Algorithm selection can drastically reduce the computational effort of some steps. This work proposes a legalization algorithm selection framework using deep convolutional neural networks. To extract the features, snapshots of circuit placements simulating three different stages of the physical design flow were used. Then, the models were trained using transfer learning, relying on pre-trained weights of the Squeezenet architecture.The adoption of transfer learning results in significant reduction in training time and required data even though the pre-trained weights come from a different problem. An extensive experimental analysis of machine learning models was performed, providing details on how the model parameters were chosen, including the convolutional neural network architecture definition, size of the images, the learning rate, and the number of epochs. The proposed framework was evaluated by training a model to select between different legalization algorithms according to two metrics, cell displacement and wirelength variation, and considering three different cell movement scenarios. The trained models achieved F-scores ranging from 0.90 to 1.00 when predicting cell displacement and ranging from 0.81 to 0.97 for wirelength variation, showing that the model was accurate even in the presence of unbalanced data. Also, the trained models outperform a non-convolutional model for both target metrics. The proposed convolutional neural network models were used to build a legalizaiton algorithm selector. When integrated into the physical design flow, the proposed algorithm selector achieved the best results for the majority of the circuits, being better than any individual legalization algorithm. Even when it does not reach the best results, it leads to results that are at most 7% worse than the best ones, showing that the model can be used as an accurate selector without compromising the flow quality. 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.

Keywords: Electronic Design Automation (EDA). Algorithm selection. Machine Learning (ML). Convolutional Neural Networks (CCNs). Physical design. Multi-row legalization.

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

$*$

LIST OF SYMBOLS

B	Circuit boundaries
C	Set of circuit cells
c_i	Cell i of a circuit
$d(c_i)$	Dimensions of cell i
FP	False positive
FN	False negative
H_{row}	Height of the circuit rows
$h(c_i)$	Height of cell i
$loc(p)$	Location of Pin p
$l(c_i)$	Location of cell i
P	Tuple defining circuit placement
$\mathcal{R}$	Set of rows of a circuit
r_i	Row i of a circuit
$\mathcal{S}$	Set of sites of a circuit
s_i	Site i of a circuit
TP	True positive
TN	True negative
W_{site}	Width of the circuit sites
$w(c_i)$	Width of cell i
X_{left}	Leftmost coordinate of circuit
X_{right}	Rightmost coordinate of circuit
$x(c_i)$	x coordinate of cell i
Y_{top}	Topmost coordinate of circuit
Y_{bottom}	Bottommost coordinate of circuit
$y(c_i)$	y coordinate of cell i

LIST OF ABBREVIATIONS AND ACRONYMS

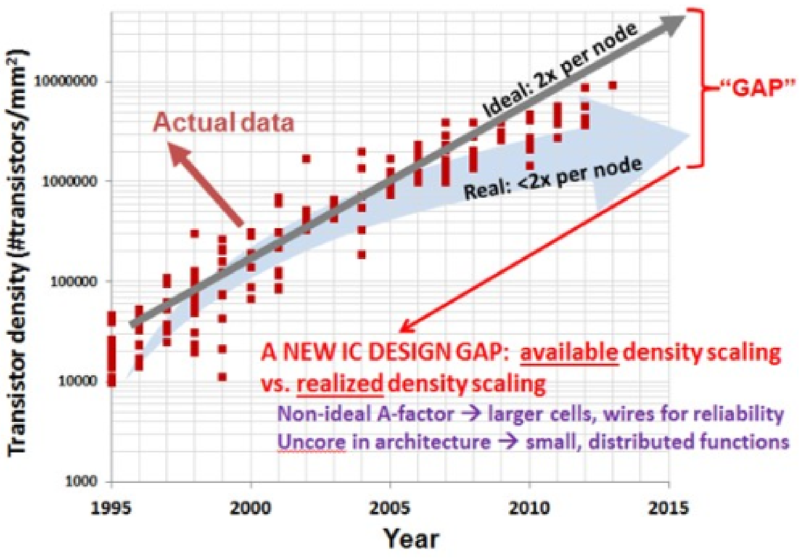
ANN	Artificial Neural Network	21, 22, 35, 73, 79
CMDP	cell movement simulating detailed placement	49, 52, 64, 65, 66, 68, 69, 71, 73
CMGP	cell movement simulating global placement	49, 50, 51, 64, 66, 68, 69, 73, 74
CMITDP	cell movement simulating incremental timing-driven placement	49, 52, 68, 71, 72, 73
CNN	Convolutional Neural Network	19, 20, 21, 22, 23, 36, 37, 39, 40, 45, 46, 48, 50, 54, 58, 59, 64, 68, 72, 73, 74
CTS	clock tree synthesis	33, 34
DRV	Detailed Routing Violation	35, 36, 37
EDA	Electronic Design Automation	18
GAN	Generative Adversarial Network	36
GBA	Graph-based Analysis	34, 35
HDL	Hardware Description Language	24
ILP	Integer Linear Programming	29
ITDP	Incremental Timing-Driven Placement	17, 41, 68
LP	Linear Programming	30
ML	Machine Learning	18, 19, 20, 21, 33, 34, 35, 36, 37, 38, 50, 73, 74
PBA	Path-based Analysis	34, 35
PPA	power, performance and area	18, 34
QP	Quadratic Programming	29, 30
SI	Signal Integrity	34
SML	Supervised Machine Learning	21, 39
VLSI	Very Large-Scale Integrated	18

CONTENTS

1 INTRODUCTION

The advancements in Very Large-Scale Integrated (VLSI) circuit (or chip) fabrication process have enabled designers to continuously increase the number of transistors in a chip as predicted by Moore's law (KAHNG et al., 2011). However, as the transistor density increases, the challenges of designing chips also grow. Electronic Design Automation (EDA) tools need to handle these challenges. The cost of handling these challenges becomes increasingly higher with higher transistor density. As a consequence, the 2013 ITRS roadmap highlighted a gap between the available transistor density provided by the fabrication process and the actual transistor density achieved by modern designs. Figure 1 presents this gap. As it can be seen, the real transistor density is less than 2x per node, which shows that designers are not able to fully benefit from the integration capability offered by advanced CMOS fabrication technologies (IEEE, 2015; KAHNG, 2018). Even though the data in the Figure is from 2013, this trend remains relevant to date.

Figure 1 – Transistor density (y axis) per year (x axis) showing the design capability gap.



Source: Kahng (2018).

The unpredictability of the design flow requires designers to iterate through the flow multiple times until the circuit meets the power, performance and area (PPA) requirements for the design. This increases the cost of designing a chip, as more work is required from engineers to be able to close the design. This might also reduce the potential PPA quality of the chip, as the designers need to do pessimistic estimations in earlier steps to avoid compromising latter steps (KAHNG, 2018). Therefore, it is important to improve predictability of the physical design flow, as this can improve the quality and reduce the cost of the design. Machine Learning (ML) is a tool that can help in this regard. A good ML model is able to predict new data from previously observed data.

ML models have been used in physical design to increase the predictability of different physical design steps, such as clock tree synthesis algorithms (KAHNG; LIN; NATH, 2013), circuit timing (HAN et al., 2014; KAHNG; LUO; NATH, 2015; KAHNG; MALLAPPA; SAUL, 2018; BARBOZA et al., 2019) and routing violations (ZHOU et al., 2015; CHAN et al., 2017; TABRIZI et al., 2018; XIE et al., 2018; YU et al., 2019; ZHOU et al., 2019; GANDHI et al., 2019; LIANG et al., 2020; YU et al., 2020; HUNG et al., 2020). 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 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 as 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, I have developed a framework to select a legalization algorithm that best fits the situation, thus increasing the quality of the outcome and avoiding prohibitively long execution times.

1.1 THESIS HYPOTHESIS

Recently, Convolutional Neural Networks (CNNs) have become popular specially for image data analysis, as this neural network architecture is capable of identifying spatial patterns in the data. Since the placement of a circuit has inherent spatial properties, I claim that CNN is a suitable model to predict features using placement data as input. Therefore, this thesis hypothesis is that using deep CNNs can improve the quality of placement techniques by selecting the best legalization algorithm for a given placement in terms of cell displacement and wirelength.

Training deep CNNs is a process that requires providing a lot of data samples to the model so that it can identify patterns on the data that can be used to predict their classes. So, in this work I'm going to adapt an existing CNN architecture and use transfer learning to reduce training complexity.

It is hard to adapt the previous works on machine learning to the legalization problem for two reasons: (1) when training the ML models, most of them use features that are specific for the problems they are tackling, which are not useful for other problems; (2) they do not provide many details of the training process, making it difficult to adapt their methodology to other works. Therefore, I am going to use an existing CNN architecture used in the ImageNet dataset (STANFORD VISION LAB, 2016), which does not use data from VLSI designs, but the model is publicly available.

I am going to adapt this existing architecture to use placement data, which is relevant for legalization. The placement data will be extraced as an image so that it can be used by the existing CNN model. Also, by using images as input data, it is possible to easily change the model to predict results for other design metrics. If we want to use the model to predict another metric in the future, we just need relabel the data based on this metric and retrain the model, without having to create more data. Finally, I am also going to provide details of our training process, so that our work can be adapted to other works that rely on placement data.

1.2 THESIS CONTRIBUTIONS

The main contributions of this work are:

- Development of a feature extraction method for legalization that is independent of the metrics being predicted.
- Design of a CNN-based algorithm selection model for legalization. Since the proposed CNNs use images as input features, the proposed model can use the same set of features to predict different metrics.
- Implementing extensive experimentation and analysis for training deep CNN using transfer learning, including which pre-trained architecture to use.
- Development of extensive experimental validation of the proposed algorithm selection framework using state-of-the-art legalization algorithms. The framework was evaluated in scenarios that simulated legalization after global placement and after an incremental optimization.

1.3 FORMAT OF THE THESIS

The remaining chapters are organized as follows.

- Chapter 2 presents fundamental concepts on machine learning and legalization necessary to understand this work.
- Chapter 3 discusses the related work on ML models used for physical design.
- Chapter 4 presents the proposed ML framework with details on the CNN training process and its integration into a physical design flow, respectively.
- Chapter 5 shows the thesis results.
- Chapter 6 draws the research conclusions.

2 FUNDAMENTAL CONCEPTS

This chapter presents the fundamental concepts necessary to understand this work. Since this work proposes an ML framework for legalization algorithm selection, the fundamental concepts in supervised learning and legalization are reviewed. First, Section 2.1 covers the fundamental concepts on supervised learning, focusing on neural networks and how they are trained and evaluated. Then, Section 2.3 covers the fundamentals on placement legalization, presenting the state-of-the art algorithms on this area.

2.1 SUPERVISED LEARNING CONCEPTS

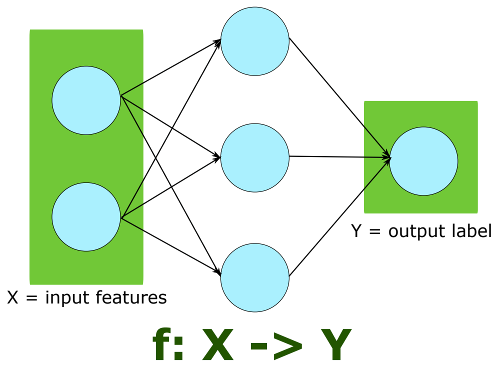
Supervised Machine Learning (SML) is the process of mapping a set of features, X, to a set of labels, Y. To do so, a function $f : X \rightarrow Y$ that maps each feature $x \in X$ to a label $y \in Y$ as accurately as possible is determined (RUSSELL; NORVIG, 2016). In SML, data is divided into two sets, the **training set** and the **validation set**. The training set is used to develop the model whereas the validation set is used to examine its efficacy.

Different models can be used for SML, such as: Decision Trees, Artificial Neural Networks (ANNs), Support Vector Machines, Random Forests, and CNNs. The framework proposed in this thesis makes use of CNNs (BISHOP, 2006; GOODFELLOW; BENGIO; COURVILLE, 2016).

To understand how CNNs work, which are a sub-case of ANNs, it is useful to first understand how an ANN work first. Figure 2 shows an example of an ANN. The neural network is composed of a sequence of node layers. The first one is the input layer, which receives the input features x. Each node applies a non-linear function to the data it receives, and passes it forward to all the nodes in the next layer. Since each node is connected to all nodes in the subsequent layer, this is called a fully connected layer. After a sequence of intermediate layers, the last one is the output layer, which generates the predicted label y. By applying a non-linear function on each layer, the neural network is capable of learning non-linear functions, which makes it a powerful SML model.

The main difference between ANNs and CNNs is the type of data they are specialized to receive as input. CNNs are specialized in spatial data as images, with one example being presented in Figure 3. Similarly to ANNs, CNNs are also composed of a sequence of layers. However, to make it possible for CNNs to explore the spatial properties of images, their initial layers are not fully connected as in ANNs. Instead, in the CNNs' initial layers a node in a layer is connected only to nodes that are spatially close to it in the next layer, and the function applied to each node is a convolution. Therefore, the CNNs' initial layers are called convolutional layers. The convolution acts as a filter in a region of pixels, to identify patterns on the image. After a series of convolutional layers, a few fully connected layers are added to predict the output label. This way, CNNs are

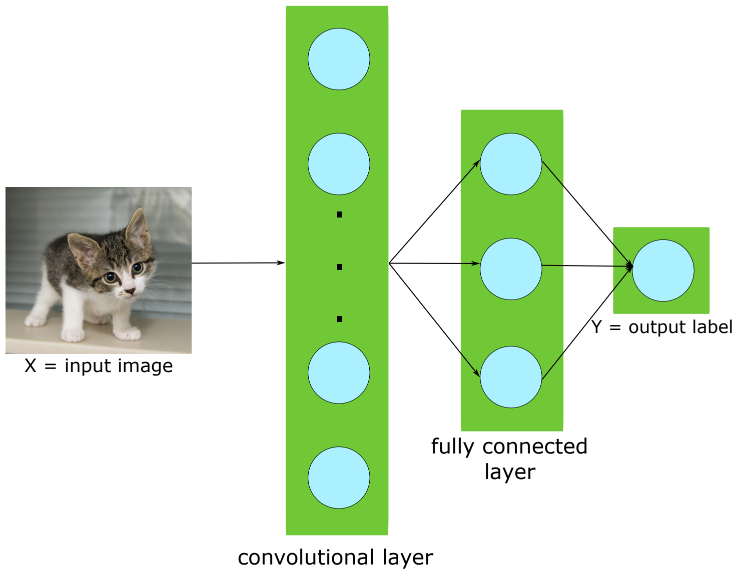
Figure 2 – Example of an artificial neural network.



Source: the author.

capable of identifying patterns in images, making them suitable for problems that have spatial data such as the legalization problem.

Figure 3 – Example of a convolutional neural network.

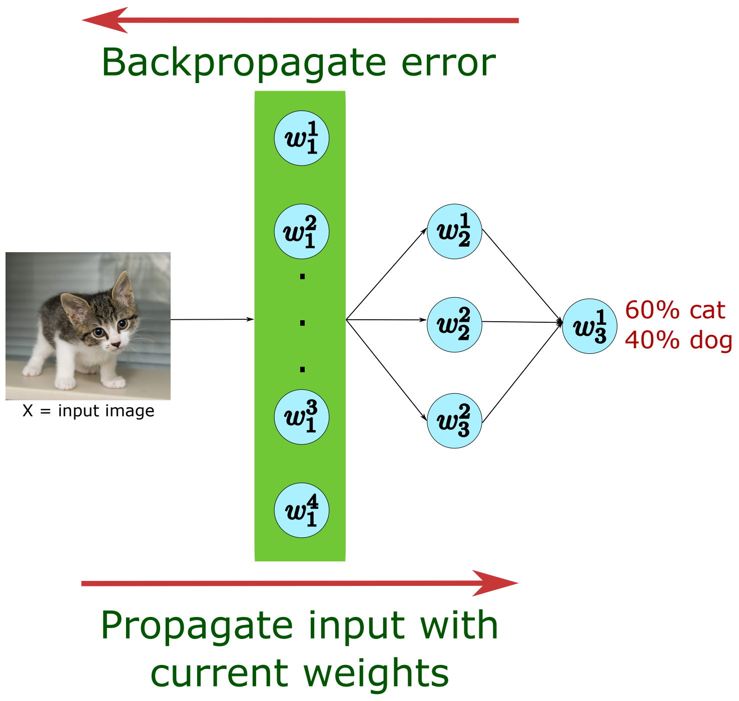


Source: the author.

In both ANNs and CNNs, the training is done using the back propagation method (GOODFELLOW; BENGIO; COURVILLE, 2016). Figure 4 illustrates how this method works. All nodes in the network are initialized with random weights. In Figure 4, w_j^i refers to the i_{th} weight of layer j. Then, for each training sample, the sample features are propagated through the network while applying the network operations in each node to get the prediction in the output layer. Then, given the real label of this sample, it is possible to calculate how big is the prediction error. This error is then backpropagated through the nodes to adjust the weights accordingly.

Applying the backpropagation method for all the training samples once is called an epoch. To improve the accuracy of the neural network, a few epochs are typically used, each epoch using the same samples applied in a different random order, so that the network does not learn a specific order.

Figure 4 – Example of ANN training process using the backpropagation method.



Source: the author.

Since CNNs typically have several intermediate 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 of using weights from a previous training process instead of initializing the network with random weights. This reduces the training time, because it is not necessary to train the network from scratch, but rather adjust the weights to fit the current data. To help using transfer learning, a few pre-trained neural networks are publicly available, such as AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), DenseNet (HUANG et al., 2017), ResNet (HE et al., 2016), SqueezeNet (IANDOLA et al., 2016) and VGGNet (KAREN; ZISSERMAN, 2014).

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. This does not reduce the training time, but reduces the amount of data necessary to train the model and, therefore, reduces the necessary amount of work for training. 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 compose more sophisticated metrics for models with imbalanced data (SASAKI, 2007):

- **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 ratio between true positive samples and the total number of samples classified as true.
- **Recall (R)**, $\frac{TP}{TP+FN}$: is the ratio between true positive samples and all the positive samples in the data.
- **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 (BISHOP, 2006). 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 be generalized 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 (EVERITT, 2006).

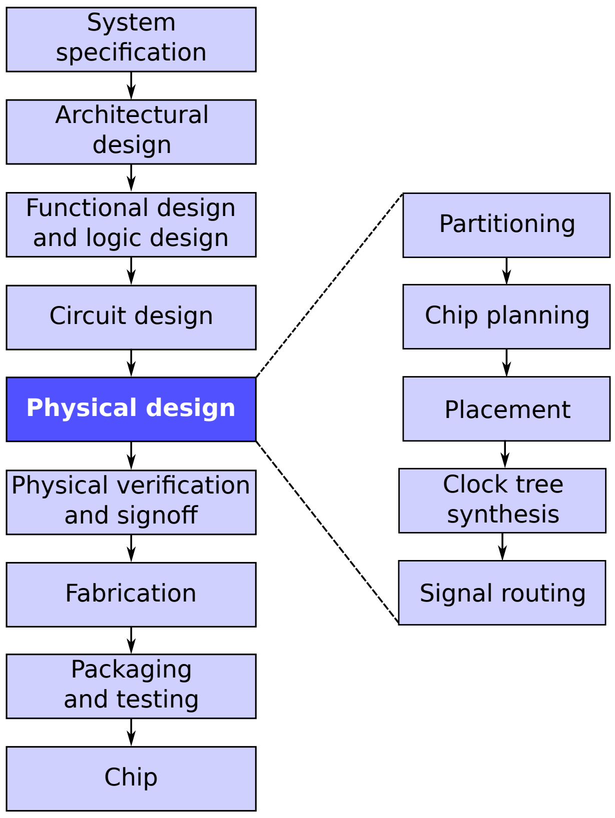
2.2 VLSI DESIGN FLOW

To cope with complexity, the design of state-of-the-art chips must follow a rigorous methodology that begins with a description of the systems at high levels of abstraction and undergoes several steps in which more details are added to enable chip fabrication. The sequence of steps is organized as the VLSI Design flow and makes intensive use of EDA tools (KAHNG et al., 2011). Figure 5 shows a simplified flowchart of this flow, starting with the system specification step, where designers decide on the chip goals and requirements. From this specification, they design the architecture in the next step by discussing things such as: analog and mixed-signal integration, types of computational cores, protocols used for input/output operations and so on. Then, in the third step, called functional design and logic design, designers describe the circuit logic using a Hardware Description Language (HDL).

Given the circuit logic description, the circuit design step makes use of logic synthesis tools to convert this description into a gate level netlist. At this point, the design flow uses a pre-characterized cell library describing logic, physical and electric characteristics of all the combinational and sequential elements available to the designer. This library is called standard cell library, and is one of the key elements to allow the semi-automated flow of chips with billions of transistors. After the circuit design step, the physical design step is responsible for mapping the logic components to physical components, placing all of the cells in the circuit area and properly connecting the net segments. The physical design step is the focus of this thesis, so its steps will be further detailed later in this chapter. After a circuit is placed and routed, the physical verification step is necessary to ensure correct logic and electrical functionality.

Once the circuit passes physical verification, it can go to fabrication, where the chips are manufactured in silicon wafers. Finally, after manufacturing, they are separated by cutting the wafer in smaller pieces to be packaged and tested. During this last testing process, some chips may be defective, due to the variance of the fabrication process. The chips that pass this last testing step are considered functional chips and can be shipped to consumers.

Figure 5 – Simplified flowchart of the standard cell design flow.



Source: Adapted from Kahng et al. (2011).

The focus of this thesis is the physical design step which is typically divided in several sub steps. These steps are partitioning, chip planning, placement, clock tree synthesis and signal routing (KAHNG et al., 2011) as depicted in Figure 5. The goal is to divide the problem into smaller sub problems that are easier to solve. The partitioning step starts by dividing the circuit into smaller modules to reduce the complexity of the next steps. The chip planning step, is responsible for arranging these modules, placing them in the chip area and defining the input/output port locations. Then, the placement step assigns physical locations to all cells inside each module. The clock tree synthesis step is responsible for connecting the clock source to all the sequential elements in each module, while the signal routing step allocates routing resources for all the circuit connections, specifying which track will be used for each net segment. In modern circuits, it is not possible to to disregard the impact of the solution of one of these steps in other steps. For example, depending on the placement solution, it might not be possible to route the circuit. Even though, modern EDA tools try to integrate these steps as much as possible, works in the literature typically handle each step separately, making it necessary to redo at least part of a previous step, in order to fix the problems in the current step (ALPERT; MEHTA; SAPATNEKAR, 2008).

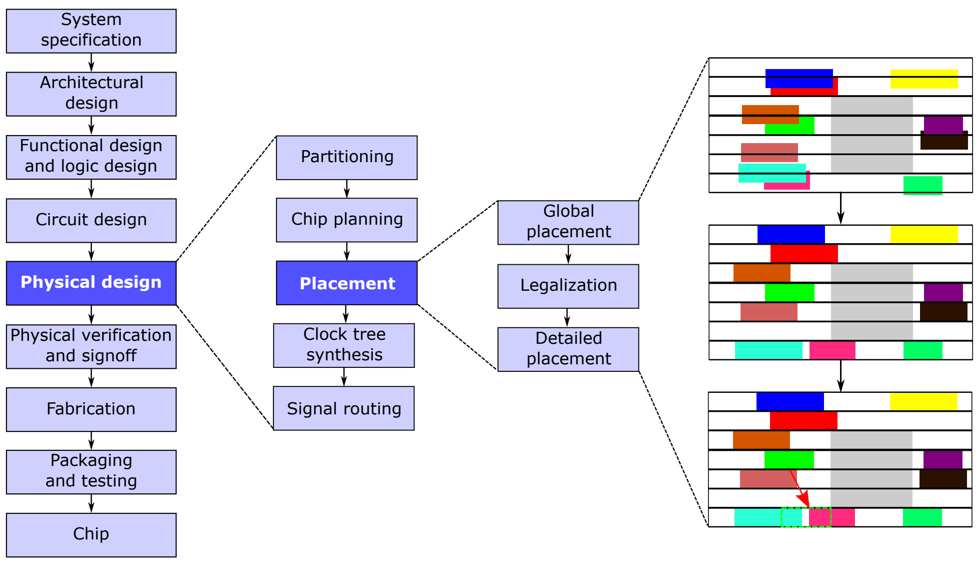
2.3 PLACEMENT LEGALIZATION CONCEPTS

The layout of an integrated circuit is made up of cells which must be placed in a two dimensional area in such a way that there are no overlaps between them. In addition, the circuit area is organized in rows and sites to which all cells must be aligned, and bounded by coordinates $B = (X_{left}, X_{right}, Y_{bottom}, Y_{top})$. These rows and sites are defined by sets $\mathcal{R} = \{r_1, r_2, ..., r_n\}$ and $\mathcal{S} = \{s_1, s_2, ..., s_n\}$, respectively. Each row r_i and site s_j have their own dimensions. For the sake of simplification, legalization algorithms often assume that all rows have the same height H_{row}, and all sites have the same width W_{site}. The set of cells is denoted as $C = \{c_1, c_2, ...c_n\}$. Each cell c_i has a location $l(c_i) = (x(c_i), y(c_i))$ and dimensions $d(c_i) = (w(c_i), h(c_i))$. Finally, a circuit placement is defined by the tuple $P = (B, \mathcal{R}, \mathcal{S}, C)$.

During the physical design flow, the placement step is responsible for assigning physical locations to all circuit cells. Due to the high number of cells in modern circuits, this step starts with a global placement which assigns these locations while ignoring the cells' dimensions. Then, a legalization step is executed to remove cell overlaps and align them with the circuit sites and rows. After legalization, a detailed placement is typically applied to refine the placement according to different metrics, such as wirelength and density. These steps can be observed in the flowchart of Figure 6. In addition, in later incremental optimization steps, such as gate sizing and incremental timing-driven placement (LIVRAMENTO et al., 2016), it may be necessary to apply placement transformations to further refine the solution. When this happens, it is necessary to execute the legalization again to ensure there are no overlaps or misalignment. Therefore, it is important to note that legalization is applied not only after the global placement, but also after any changes are done in the placement (NETTO et al., 2017).

In Figure 7 (a), an illegal placement of a circuit with 11 cells (c_1 to c_{11}), repre-

Figure 6 – Simplified flowchart of the placement steps in the standard cell design flow.



Source: Adapted from Kahng et al. (2011).

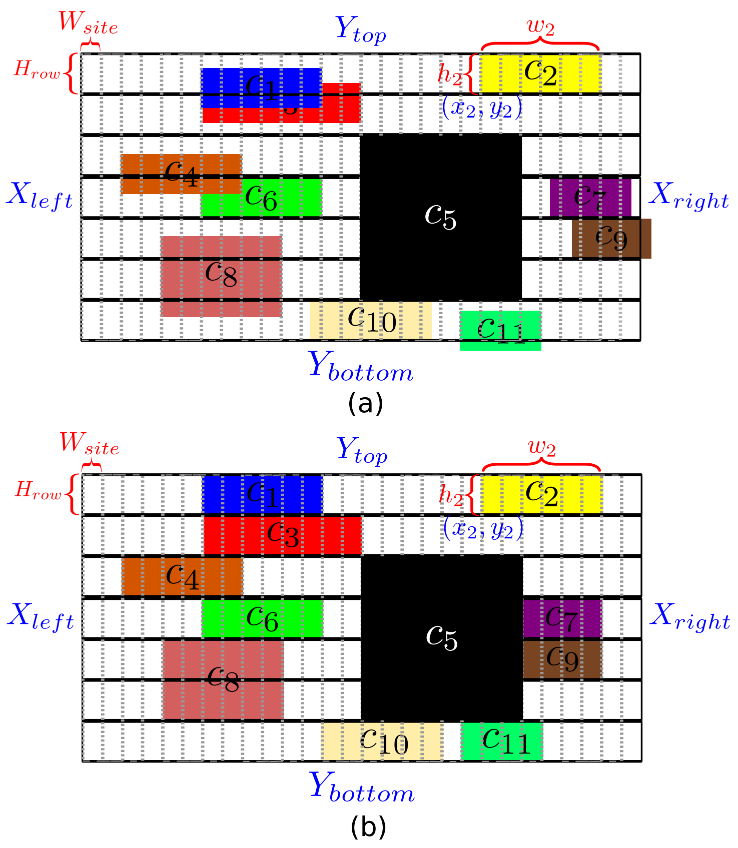
sented by colored rectangles, is illustrated. The black cell (c_5) is larger than the other ones because it represents a fixed macroblock, which cannot be moved during legalization. In this example, it is possible to observe three types of placement violations: 1) cell overlaps (c_1 and c_3, c_4 and c_6); 2) cells not aligned with circuit sites and rows (c_7, c_8, c_9 and c_{10}); 3) cells outside the circuit boundaries (c_9 and c_{11}). A legalization algorithm that receives this placement as input shall move a subset of cells to remove all violations, thus producing a legal placement as the one in Figure 7 (b).

The legality constraints can be formalized as follows: a placement P is legal if it satisfies the constraints in Equations (2.1) to (2.6). The constraints in Equations (2.1) to (2.4) are defined for each cell $c_i \in C$ and ensure that all cells lie inside the circuit boundaries and are properly aligned to circuit sites and rows. Besides that, each pair of cells $x(c_k)$ and $x(c_j)$ in the same row must satisfy Equation (2.5) which avoids overlaps between them in the x axis. A similar constraint must be met for pairs of cells $y(c_k)$ and $y(c_j)$ in the same column, as defined in Equation (2.6).

$$X_{left} \leq x(c_i) \leq X_{right} - w(c_i) \tag{2.1}$$
$$Y_{bottom} \leq y(c_i) \leq Y_{top} - h(c_i) \tag{2.2}$$
$$x(c_i) = n \times W_{site}, n \in \mathbb{N} \tag{2.3}$$
$$y(c_i) = m \times H_{row}, m \in \mathbb{N} \tag{2.4}$$
$$(x(c_k) + w(c_k) < x(c_j)) \vee (x(c_j) + w(c_j) < x(c_k)) \tag{2.5}$$
$$(y(c_k) + h(c_k) < y(c_j)) \vee (y(c_j) + h(c_j) < y(c_k)) \tag{2.6}$$

Given an illegal placement P, a legalization algorithm must produce a new placement P' that satisfies the circuit legality constraints. Placement P' is obtained by re-

Figure 7 – Example of illegal placement (a) and legal placement (b) of a digital circuit with 11 cells



Source: the author.

locating a subset of cells C' while minimizing a cost function. Equation 2.7 describes a commonly used cost function, which measures the placement perturbation as the Manhattan distance[1] between the cell location before and after the legalization.

$$cost(\mathcal{L}) = \sum_{c_i \in C} |x(c_i) - x'(c_i)| + |y(c_i) - y'(c_i)| \tag{2.7}$$

Modern designs include some additional challenges. For example, in modern circuits it is common to have cells with different heights, which are called multirow cells. This increases the complexity of legalization because traditional algorithms legalize all circuit rows independently. With cells that span throughout multiple rows this is not possible anymore.

Another challenge is the existence of fence regions. Modern circuits have different voltage islands to achieve low power constraints. This must be handled in the placement by having separate regions for the voltage islands. Those regions are called fence regions, and they impose constraints that some cells can only be placed inside specific fence regions, and no other cells can be placed there.

Finally, the placement of the cells can affect the possibility of routing the nets later in the design flow. For example, depending on where some cells are placed, they

[1] Given two points (x, y) and (x', y'), the Manhattan distance is given by $|x - x'| + |y - y'|$.

may block the access to some pins. Those are not placement violations, but they will result in routing violations, being referred to as pin short violations. Recent legalization algorithms use heuristics to avoid pin short violations.

A legal placement can be produced by using greedy heuristics (CHOW; PUI; YOUNG, 2016; LI et al., 2018; DO; WOO; KANG, 2019), dynamic programming (WANG et al., 2017), Integer Linear Programming (ILP) (HUNG; CHOU; MAK, 2017) or Quadratic Programming (QP) (CHEN et al., 2017; ZHU et al., 2020).

Among the existing legalization algorithms, three of them represent the state-of-the-art in legalization: Li et al. (2018), Zhu et al. (2020) and Do, Woo & Kang (2019). All three algorithms can handle multirow cells, fence regions and consider maximum cell displacement in the optimization target. Because of that, this thesis focuses on these algorithms, so they are described in the following sections.

2.3.1 Li et al. (2018)

The algorithm from Li et al. (2018) has three steps: multirow global legalization, maximum displacement optimization and cell shifting. The first step is an improvement of the work from Chow, Pui & Young (2016) and legalizes each cell sequentially using a greedy heuristic. For each target cell c_i to be legalized, a local window is created around the global placement location of the cell. Only cells inside this local window are considered when legalizing c_i. Then, the algorithm finds a set of insertion points where c_i can be legally placed only by shifting cells in the local window, but without changing their order or y coordinates. For each insertion point p_i, the displacement curve is evaluated considering the c_i and all cells inside the local window, in order to find the optimal x coordinate to place c_i in p_i. This process is repeated for all insertion points and then c_i is placed in the optimal location. After c_i is placed, its y coordinate and order in the circuit rows are fixed.

Since the cells are legalized in sequential order in the first step, it is possible that the last cells of the ordering have a large displacement, increasing the maximum displacement of the solution. Because of that, the second step of the algorithm aims to reduce maximum displacement without compromising total displacement. This is done by using a bipartite graph matching strategy. For a subset of cells of the same type (i.e. instances of the same element from the standard cells library), a bipartite graph is built where one partition represents the cells and the other represents their locations. Each cell can be placed in any location, so there is an edge between every pair of nodes in the two partitions. The cost of each edge is proportional to the displacement resulting by moving a given cell to a target location, while penalizing overly large displacements. The goal is to assign cells to legal locations while considering both total and maximum displacements. In the end, the match is found by solving a min-cost flow problem.

The last step of the algorithm aims to further optimize cells' locations without changing their y coordinates or order inside their rows, which means cells can only be shifted horizontally in their rows. This is done with a Linear Programming (LP) formulation that minimizes total and maximum displacement with constraints to avoid overlaps between cells. Instead of solving this LP problem, the authors solve the dual problem as a min-cost flow problem.

In all three steps of the technique, the authors considered pin short violations. To avoid pin short violations, insertion points that overlap with power and ground rails or IO pins are avoided in the multirow global legalization, and in the third step of the algorithm the feasible region of each cell is limited to avoid violations. When performing the bipartite graph matching optimization, no violations are inserted because cells are only swapped in pairs of the same type.

2.3.2 Zhu et al. (2020)

The algorithm from Zhu et al. (2020) is divided into four steps: fence region handling, movement-aware cell reassignment, technology-aware legalization and technology-aware refinement. The first step is simple and only moves cells to their assigned fence regions. Cells that do not belong to any fence region are assigned to a default region. In both cases, each cell is aligned to its nearest row that satisfies the following constraints: 1) the row belongs to the cell fence region; 2) the cell does not overlap with any macroblock when placed in this row; 3) the cell should meet the power and ground alignment constraints in this row; 4) the cell should not create a pin short violation with relation to the power and ground rails.

After assigning cells to their fence regions, the movement-aware cell reassignment step aims to remove cell overlaps. This is done by formulating the legalization problem as a relaxed QP problem. The proposed formulation assigns x and y coordinates for each cell while minimizing the displacement. In order to simplify the problem, it is relaxed by ignoring the right boundary of the circuit. After that, cells in dense regions might still have large displacements, so the algorithm reassigns these disruptive cells to better locations. This is done by identifying all disruptive cells, finding target regions for them, and then assigning disruptive cells to regions using a min-cost flow formulation. In order to avoid large displacement for a single cell assignment, a region is considered as a valid target if a cell can be placed there while generating movement only in other cells that are below a specified threshold.

Since the second step legalizes cells using a relaxed problem formulation, there might be some illegal cells. The technology-aware legalization aims to remove these violations. Cells that are not in legal locations are moved to the closest violation-free location, if this location does not violate a specified displacement threshold. Otherwise, the algorithm extracts blank spaces around the cell original location in order to find a place where the cell can be legalized with lowest movement. In the end of this third step, all cells should be in legal locations.

The technology-aware refinement step aims to further optimize displacement. First, it reduces maximum displacement by identifying cells with large displacement and swapping its location with another cell. Two cells can swap if this does not violate any legality constraints, if it does not increase the number of technology constraints violations and if the maximum displacement is reduced. Finally, it reduces total displacement by swapping cells in local regions. The circuit is divided into bins and a bipartite graph is built for each region. This bipartite graph is similar to the one used by Li et al. (2018), where one partition corresponds to the cells and the other one corresponds to their locations. Then, cells are assigned to optimal locations by solving this bipartite graph matching problem.

2.3.3 Do, Woo & Kang (2019)

The algorithm from Do, Woo & Kang (2019) has three steps: pre-legalization, multirow cell legalization and quality refinement. The first step legalizes only the cells that are outside their fence regions. This is done by moving them to the closest location inside their respective fence regions. Then, their locations are fixed. This is done because, according to the authors, these cells can result in large displacement if they are left to be legalized later, since they are already outside the fence region they should be. Therefore, their technique prioritizes them to avoid a large impact on maximum displacement.

The second step legalizes the remaining cells of the circuit. This is done in a sequential way and the authors use three different techniques. For each cell c_i, the first technique runs a breadth-first search to find the nearest available region for c_i. If the algorithm cannot find an available region within a search limit, it proceeds to the second technique. The second technique selects a target location for c_i and shifts the cells in a small window around this location to avoid overlaps. If it is still not possible to place c_i in the target location, the algorithm tries the third technique, called corner weight move. This final strategy divides the fence regions into sub-regions, and then moves the cells inside the sub-regions towards the corners. This is done to open space to place the illegal cells. The authors observed that this strategy is effective when the fence regions are densely occupied. In the end of the multirow cell legalization, all cells should be in legal locations.

The quality refinement step aims to further reduce total and maximum displacement of the circuit. This is done using a simulated annealing algorithm with two operations: cell move and cell swap. The cell move operation selects a cell c_i whose displacement is below a specified threshold and moves it to a different location. This might increase the displacement, but it can reduce the total displacement, as the cells around the previous location of c_i can also be moved to reduce their displacements. The cell swap operation selects two cells of the same size and swaps their locations. This is effective when the circuit is dense, as this operation can reduce displacement without changing the placement layout.

2.4 SUMMARY

This chapter presented the fundamental concepts that are important for understanding this thesis. I explained how neural networks work and how the backpropagation method is used to train them. I also discussed about some techniques that can be used to reduce training time, and which metrics can be used to evaluate the trained network. Besides that, I also presented the fundamental concepts on placement legalization, presenting the three state-of-the-art algorithms on this topic. In this thesis I am going to focus on these three algorithms since they present high quality results, but with no clear winner among them[2]. So, I expect the ML model to be able to learn in which situation each algorithm performs better.

[2] Chapter 5 shows how those algorithms differ between the circuits.

3 RELATED WORK IN MACHINE LEARNING FOR PHYSICAL DESIGN APPLICATIONS

In this chapter I am going to discuss the related work that use ML for physical design applications. 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. 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. The following sections on this chapter will cover the related work for each one of these steps.

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

Physical design step	Work	ML model
CTS	Kahng, Lin & Nath (2013)	non-convolutional
Timing analysis	Han et al. (2014) Kahng, Luo & Nath (2015) Kahng, Mallappa & Saul (2018) Barboza et al. (2019)	non-convolutional
Routing	Zhou et al. (2015) Chan et al. (2017)	non-convolutional
	Tabrizi et al. (2018) Xie et al. (2018) Yu et al. (2019) Zhou et al. (2019) Gandhi et al. (2019) Liang et al. (2020) Yu et al. (2020) Hung et al. (2020)	convolutional
Legalization	Netto et al. (2019) This work	convolutional

3.1 RELATED WORK IN CLOCK TREE SYNTHESIS

Kahng, Lin & Nath (2013) have proposed one of the first works to use ML models in physical design. They showed that changing clock tree parameters such as aspect ratio of the circuit, location of the clock source and other necessary parameters for clock tree synthesis (CTS) may result in variations of more than 40% in the clock delay. Without ML, the way designers have to handle this variation is by testing different parameter configurations in order to find the best combination. However, due to the large design space and long execution time to run CTS algorithms for large circuits, doing this for many parameter configurations in infeasible.

In order to circumvent this issue, Kahng, Lin & Nath (2013) trained regression models to predict clock tree delay given several parameters such as: number of sinks, maximum skew, maximum insertion delay, block area, percentage of block area, maximum buffer size, among others. They generated artificial and realistic circuits to train different types of ML models. After trained the models, they evaluated how well the model can guide a CTS flow towards better parameters. The best models achieved a worst-case estimation error of 13%.

3.2 RELATED WORK IN TIMING ANALYSIS

One of the challenges in achieving timing closure in the physical design flow is that all steps impact on timing but it is too time consuming to accurately estimate timing in early stages. For example, after placement it is possible to know the location of cell pins, however the location of the routing segments is still unknown, which makes timing less accurate. Even when this information is available, running an accurate timing engine (called signoff timing engine) is too time consuming to be executed several times in the flow, so designers use less accurate timing engines (called design implementation engine) that assume some simplifications to quickly estimate timing. Because of that, there is a miscorrelation between the estimated timing during the physical design flow and the actual circuit timing, which may require more iterations in the flow until achieving timing closure. Because of that, a few works evaluate different ML models to reduce these miscorrelations.

The work of Han et al. (HAN et al., 2014) focuses on predicting the miscorrelation between two different design implementation engines and between a design implementation engine and a signoff timing engine. They observed that the miscorrelation between different engines can reach more than 100 ps, and trained regression models to predict the result of another engine given a reference one. In order to train the models they extracted different features that represent timing information such as capacitance, resistance, slew and delay of cells and wires. As a result, they managed to reduce the miscorrelation between the engines to less than 30 ps.

One limitation of the work from Han et al. (2014) is that it works only for engines with the same configuration in terms of Signal Integrity (SI) mode, which means that either tools should be in SI mode or neither of them to properly reduce the miscorrelations. To cope with that, Kahng, Luo & Nath (2015) trained models to predict SI timing from non-SI timing. This way, it is possible to run the engine in non-SI mode, which is faster but less accurate, and obtain SI-mode results. Besides the features used by Han et al. (2014), Kahng, Luo & Nath (2015) also extracted other features related to SI and non-SI timing. As a result, their best models reduced worst-case error in incremental delay prediction from 60ps to 5.2ps.

Another issue that may degrade design PPA is pessimistic timing estimations. For example, typically in the design flow timing is estimated using Graph-based Analysis (GBA), which is much faster than Path-based Analysis (PBA), but it is more pessimistic, as it does not consider path information when propagating slew and delays. Because of that, the actual path delay may be lower than the one estimated by the timing engine.

To reduce the impact of this pessimistic estimation, Kahng, Mallappa & Saul (2018) proposed an ML model to predict path-based slack from graph-based results. The authors extracted 13 features from GBA such as transition time, arrival times, drive strengths, cell functionality, cell fanout, load capacitance and propagation delay. From these features, they trained regression models and managed to reduce the divergences between the GBA and PBA from 9.43 ps to 6.06 ps on average.

Finally, the most recent work on machine learning for timing prediction is the work of Barboza et al. (2019). This work is similar to Han et al. (2014) as they also focus on predicting signoff timing using less accurate timing information. However, the main difference in Barboza et al. (2019) the authors goal is to preform this prediction in earlier physical design steps, so they used only pre-routing features, such as driver and sink capacitance, distance between driver and target sink, maximum driver input slew and context sink locations. They evaluated different regression models and observed that even without routing information, a random forest was able to achieve a correlation of 0.971 with respect to the signoff tool, and it was more effective in correctly identifying the critical paths than the estimation of a design implementation tool.

3.3 RELATED WORK IN ROUTING

The earliest work on ML for routing prediction used non-convolutional models to identify Detailed Routing Violations (DRVs) during global routing, as the three ones described in the sequel. Traditionally, global routing algorithms used congestion maps to estimate where DRVs will happen and guide the routing process with this estimation. However, there is a high miscorrelation between the congestion map and the locations of the actual detailed routing violations. Looking to overcome this problem, Zhou et al. (2015) proposed an ML model to predict the number of DRVs in a global routing solution. They extracted features such as pin density, routing blockages and net information to train a model that achieves average accuracy of 83%.

Chan et al. (2017) further improved this work by using an ML model to build a DRV hotspot map during global routing. Their idea is to identify the occurrence of DRVs in different areas of the routing using similar features than Zhou et al. (2015). Then, they proposed a routability optimization algorithm that makes use of this hotspot map to reduce the risk of DRVs during global routing. Their results showed that the proposed optimization reduced the number of DRVs by 20% on average.

The last work to use non-convolutional model for routability predict was the work from Tabrizi et al. (2018). The main difference between this work and the previous ones is that Tabrizi et al. (2018) used an ANN and focused on short violations, a specific type of DRV. They trained a classification model to identify if there will be a short violation in a given GCell. In order to do that, they used features regarding the region location in the design, cell density, routing acessibility, number of local and global nets, and others.

Their trained model was able to predict on average 90% of the short violations with only 7% of false alarm.

Other works explored the use of convolutional models to predict routing violations. Their goal is to use convolutional models to identify spatial properties in the circuit placement which can lead to routing violations. For example, the work from Xie et al. (2018) uses CNNs to predict the number of DRVs and to build a DRV hotspot map. In order to train the CNNs they extracted images representing different characteristics of the circuit, such as macro distribution, cell distribution, cell density and routing congestion. The extracted images were stacked in a single image with multiple layers and used to train the network. The authors observed that the ML model improved the global routing accuracy in identifying DRVs by 50%.

Yu et al. (2019) and Yu et al. (2020) propose a different strategy to identify short violations using CNNs. Instead of generating images from the whole circuit area, or even a circuit region, they make snapshots of small parts of the circuit in order to identify pin patterns of pairs of cells that, when placed side by side, will likely lead to short violations. By doing so, it is possible to identify these potential violations before global routing, as only the cell locations are necessary to generate the images. The authors also propose a detailed placement algorithm that makes use of this prediction to avoid cell combinations with high probability of short violations. By doing so, for a set of non-public industrial circuits, they reduced the number of short violations by 79% and the total number of DRVs by 51%, with only 1% wirelength overhead.

Going back to works on DRV violation map prediction, Hung et al. (2020) focus on using only features from the congestion report generated by global routing. For each GCell, they used Cadence's Innovus tool to extract the number of total tracks, remaining tracks, overflow tracks in the vertical and horizontal directions. A data sample is created by extracting a 5x5 GCell window, forming an image to feed the CNN. Since there are many more samples without DRVs, the authors divided the samples in GCells with DRV, GCells without DRVs and GCells surrounding DRVs. Then, they undersampled the GCells without DRVs to make the data more balanced. The trained model achieved an F-score of 0.79.

Zhou et al. (2019) and Liang et al. (2020) also used CNNs to predict DRV violation maps, but using different architectures. Zhou et al. (2019) propose using a deep convolutional Generative Adversarial Network (GAN) to generate the violation map image. The input features are the pin density, routing channel capacities and net density. Given that, the generator model has a sequence of encoding layers to extract information from the input features and a sequence of decoding layers to generate a new image representing the violation map. The model achieves a mean absolute error of at most 4.47%. Liang et al. (2020) use a similar approach, but they propose a customized J-net architecture. Some of the differences are that input channels of different resolutions are fed into different levels of the enconding layers and the number of decoding layers is lower than the encoding ones. As features they used pin configuration images and tile-based feature maps. Results showed that their model improved true positive rate by up to 40% when compared to other recent works.

Finally, Gandhi et al. (2019) propose a different approach on the usage of ML for routing improvement. Instead of predicting DRVs, they proposed a reinforcement learning model to actually route the circuit. Their proposed model is based on AlphaGo Zero, and they modeled the routing problem as a min-max game where a router generates the net segments, and a cleaner removes segments with routing violations. These two agents take turns until all the circuit is routed. So far, their technique has been validated on small 5x5 routing grids.

3.4 RELATED WORK ON LEGALIZATION

The only related work that focuses on predicting the quality of legalization algorithms is our previous work (NETTO et al., 2019). In that work, an ML model was trained 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.

However, that work used a non-convolutional model with the following features: density of the circuit, area occupied by cells of each height, histogram of area occupied by cells on each row, histogram of area occupied by overlaps on each row.

3.5 SUMMARY

In this chapter I presented the main related work concerning this thesis. It is worth noting four characteristics of the related work. First, 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 these models. In this work I propose using CNNs since I am using placement data which has inherent spatial features.

Second, only Kahng et al. (KAHNG; LIN; NATH, 2013) 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 Guo & Hsu (2003), Smith-Miles (2009), Kotthoff, Gent & Miguel (2012), Kerschke & Trautmann (2019), and can be used in other physical design steps as well, as legalization.

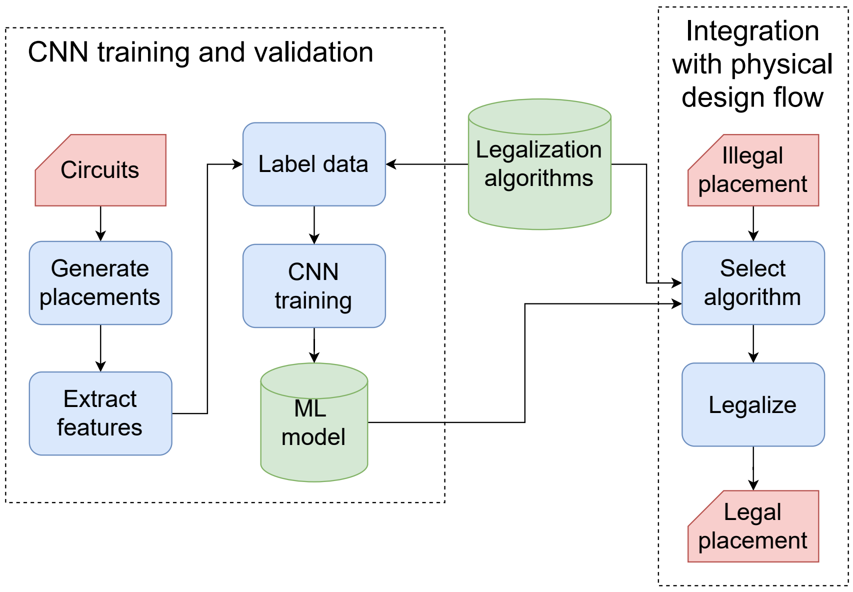
Third, most related work target timing analysis and routing, while only a few address other steps like CTS, leaving much room for investigating the application of ML to other physical design steps. Because of that, I focus this work on the legalization step, as the only work that have explored the use of ML models in this step so far is my previous work.

Fourth, 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. Therefore, in this work I use pre-trained models to perform our prediction tasks.

4 PROPOSED ALGORITHM SELECTION FRAMEWORK

In this work, I apply SML to predict the outcome of legalization during placement. Our proposed model is a classification problem where a list of legalization algorithms $\mathcal{L}$, and a placement P 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 which results in the best solution without having to run all legalization algorithms, thus saving execution time.

Figure 8 – Flowchart of the proposed algorithm selection framework



Source: the author.

Figure 8 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, I generate placements from the input circuits to increase the amount of training and validation data. This is necessary due to the limited number of circuits I have available for training the CNN. I extract features from the training data by saving the images of each placement and I label the data according to the score achieved by each legalization algorithms. Given that, I 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 I use the CNN model to predict the best legalization algorithm to use for each illegal placement. Then, I legalize the placement using the selected algorithm.

4.1 PROPOSED TRAINING DATA GENERATION

Algorithm 1 presents the proposed data generation strategy to train and validate the machine learning model.

Algorithm 1: GENERATE_DATA($\mathcal{L}_i$, P)

```
Input  : Legalization algorithm $\mathcal{L}_i$ and circuit placement $P$
Output: Training data
1  for $c_i \in C$ do
2      $l'(c_i) \leftarrow l(c_i)$;
3  end
4  $n\_samples \leftarrow 1000$, $F \leftarrow \emptyset$;
5  for $num\_sample \leftarrow 1$ to $n\_samples$ do
6      MOVE_CELLS($P$);
7      $\Lambda_i, result_i \leftarrow \mathcal{L}_i(P)$;
8      $I \leftarrow$ SAVE_IMAGE($P$);
9      $\delta_i \leftarrow$ EVALUATE_SOLUTION($\Lambda_i$, $P$);
10     $\mathcal{F} \leftarrow \mathcal{F} \cup (I, \delta_i, result_i)$;
11     for $c_i \in C$ do
12         $l(c_i) \leftarrow l'(c_i)$;
13     end
14 end
15 return $\mathcal{F}$;
```

The algorithm receives as input a set of legalization algorithms $\mathcal{L}_i$, and the circuit placement P. In the end, the algorithm outputs data from 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 design (line 4). For each sample, the algorithm moves the cells to generate a new placement (line 6) and then it legalizes the new placement (line 7). Once the placement is legal, the placement image, I, is used as an input feature, $\mathcal{F}$, of the data sample (line 8). 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 design and the 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, I 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 goal of the MOVE_CELLS function is to mimic a new input placement for the legalization algorithm. Since the legalization can be applied in different parts of the physical design flow, such as after the global placement or after an incremental optimization step, I implemented different versions of the MOVE_CELLS function. The first implementation aims to mimic a global placement solution and therefore, it moves all movable cells, as shown in Algorithm 2.

Algorithm 2: MOVE_CELLS_GP(P)

```
   Input  : Placement $P$
   Output: Set of moved cells $C'$
1  foreach $c_i \in \mathcal{C}$ do
2      if IS_MOVABLE($c_i$) then
3          $r_x \leftarrow$RANDOM(-10000, 10000);
4          $r_y \leftarrow$RANDOM(-10000, 10000);
5          $x(c_i) \leftarrow x(c_i) + r_x$;
6          $y(c_i) \leftarrow y(c_i) + r_y$;
7          $x(c_i) \leftarrow$min($X_{right} - w(c_i)$, max($X_{left}, x(c_i)$));
8          $y(c_i) \leftarrow$min($Y_{top} - h(c_i)$, max($Y_{bottom}, y(c_i)$));
9      end
10 end
```

Algorithm 2 receives as input the placement P, and randomly moves all movable cells in the circuit. Some cells, such as blockages, are fixed and cannot be moved while generating a new 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 $-10,000$ and $10,000$ placement units. The cell is moved to the new location while ensuring it remains inside the circuit bounds (lines 5–8). Observe that it is not necessary to keep cells within circuit bounds to run legalization, but I did that to keep it more similar to the initial global placement solution.

The second cell movement I implemented simulates a detailed placement technique. Since detailed placement only refines the global placement solution, it typically does not move all the cells, but only a subset of them. Also, it receives as input a legal placement. Algorithm 3 shows how this can be done.

This algorithm is very similar to the previous one, however I want to reduce the number of moved cells. In order to do that, the algorithm generates a random number r_m between 0 and 1. Then, a cell is moved in lines 3-10 only if the cell is movable and r_m is less than 0.1. This way, only about 10% of the cells are moved.

Finally, in the third type of cell movement I wanted to simulate an ITDP technique. ITDP techniques aim to improve circuit timing and because of that they move only a subset of cells that will likely improve timing. For example, one possibility is to

Algorithm 3: MOVE_CELLS_DP(P)

```
Input  : Legal placement $P$
Output: Set of moved cells $C'$
foreach $c_i \in \mathcal{C}$ do
    $r_m \leftarrow$RANDOM(0, 1);
    if $IS\_MOVABLE(c_i) \wedge r_m$ then
        $r_x \leftarrow$RANDOM(-10000, 10000);
        $r_y \leftarrow$RANDOM(-10000, 10000);
        $x(c_i) \leftarrow x(c_i) + r_x$;
        $y(c_i) \leftarrow y(c_i) + r_y$;
        $x(c_i) \leftarrow \min(X_{right} - w(c_i), \max(X_{left}, x(c_i)))$;
        $y(c_i) \leftarrow \min(Y_{top} - h(c_i), \max(Y_{bottom}, y(c_i)))$;
    end
end
```

move only the cells in the critical path, or only cells in paths with negative timing slack. However, in the benchmarks used in this work I do not have access to timing information. Because of that, I cannot find which paths have negative timing slack. In order to circumvent this issue, I used Algorithm 4 to find the circuit paths, regardless of their slacks.

Algorithm 4 receives as input a set of pins PN (including cell pins and circuit pads) and a number of circuit paths n_p to trace and outputs a set of cells C' in the circuit paths. The first step is to identify the circuit primary outputs. Therefore, all pins are visited and the primary outputs are added to a new set PO (lines 1–6). A pin is considered a primary output if it is an output pad, or an input pin of a flip-flop. After finding the primary outputs, the algorithm finds the cells in the paths by backtracking from the primary outputs.

This is done as follows for each primary output in arbitrary order: first, a list of nets is initialized with the net connected to the primary output (line 11). The net connected to a pin p_i is denoted by $n(p_i)$. Then, it starts visiting the nets in this list (lines 12–27) and for each net n_i it visits the pins connected to this net, denoted by $pins(n_i)$ (lines 14–26). For each output pin, if the pin belongs to a cell that is not already in set C', the cell is added to this set (lines 17–18). The cell connected to a pin p_i is denoted as $c(p_i)$. Finally, all the other nets connected to this cell are also added to the list of nets N (lines 19–23), so that they are visited in the next iterations. This is done by visiting all pins p_j connected to the cell, except p_i, which was already visited. The pins connected to a cell c_i are denoted by $pins(c_i)$.

After finding the cells connected to the timing path, the algorithm checks if there is more than one cell in this path. I do this because some circuit paths do not contain many cells, and I want to focus the movement on paths with several cells. So, if there are more than 1 cell in the path, the number of paths is incremented (lines 28–29). Finally, if the number of paths reached the desired number n_p, the algorithm ends returning the

Algorithm 4: FIND_CIRCUIT_PATHS(PN, n_p)

```
Input  : Set of pins $PN$, number of circuit paths $n_p$
Output: Set of cells $C'$ in the circuit paths
1  $PO \leftarrow \emptyset$;
2  foreach $p_i \in PN$ do
3      if $IS\_PRIMARY\_OUTPUT(p_i)$ then
4          $PO \leftarrow PO \cup \{p_i\}$;
5      end
6  end
7  $C' \leftarrow \emptyset$;
8  $paths \leftarrow 0$;
9  foreach $po_i \in PO$ do
10     $added\_cells \leftarrow 0$;
11     $N \leftarrow [n(po_i)]$;
12     while $N \neq \emptyset$ do
13         $n_i \leftarrow N.pop()$;
14         foreach $p_i \in pins(n_i)$ do
15             if $IS\_OUTPUT\_PIN(p_i)$ then
16                 if $c(p_i) \notin C'$ then
17                     $C' \leftarrow C' \cup \{c(p_i)\}$;
18                     $added\_cells \leftarrow added\_cells + 1$;
19                     foreach $p_j \in pins(c_i)$ do
20                         if $p_j \neq p_i$ then
21                             $N \leftarrow N \cup \{n(p_j)\}$;
22                         end
23                     end
24                 end
25             end
26         end
27     end
28     if $added\_cells > 1$ then
29         $paths \leftarrow paths + 1$;
30         if $paths == n_p$ then
31             return $C'$;
32         end
33     end
34 end
```

set of cells C' (lines 30–32).

After finding the cells in the desired number of circuit paths, all cells in these circuit paths are moved as in the previous two strategies, by calling Algorithm 2 only for the set of cells C'. However, as in the second strategy, the input here is a legal placement. Notice that in all three moving strategies, the amount of movement was between -10,000 and 10,000 placement units[1]. The benchmarks evaluated in this work have dimensions ranging from 342,000 to 900,000 placement units. Hence, the amount of random movement

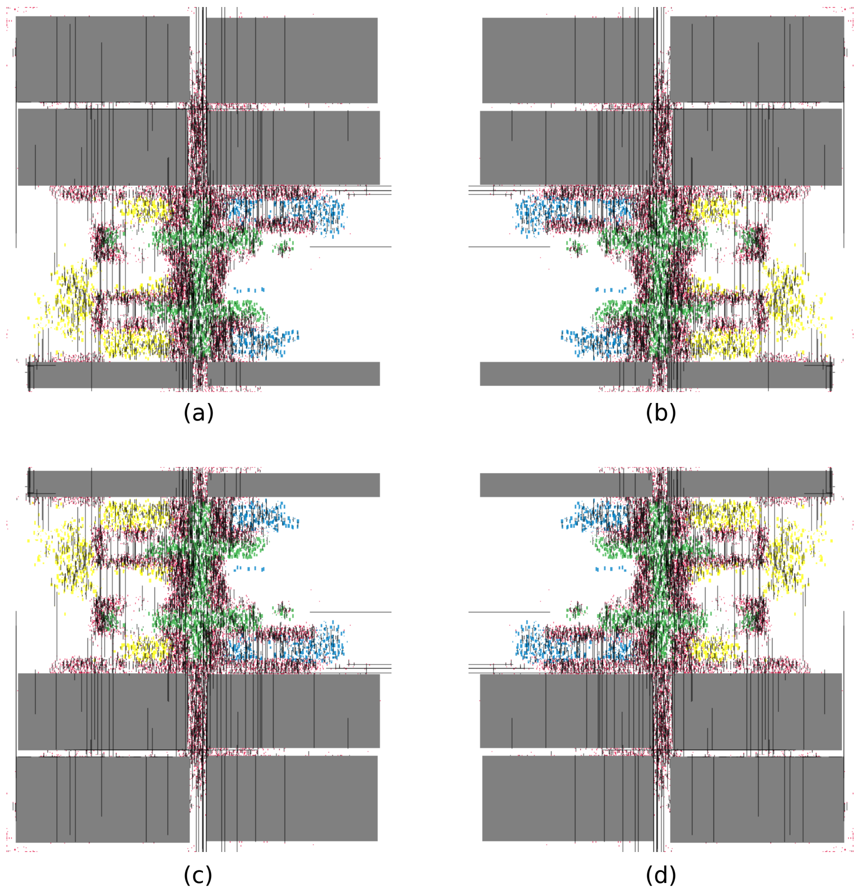
[1] The (0, 0) coordinate of the chip is the bottom left corner. So a negative movement in the x axis represents a movement to the left, while a negative movement in the y axis represents a movement to the bottom direction.

applied to each cell represents at most only 3% of the circuit dimensions, resulting in small perturbations of the initial placement.

4.2 FEATURE EXTRACTION

Features with spatial properties are better suited for CNN models. Therefore, I generate snapshot images of the circuit placements. However, I 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; 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 9.

Figure 9 – 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 axis.

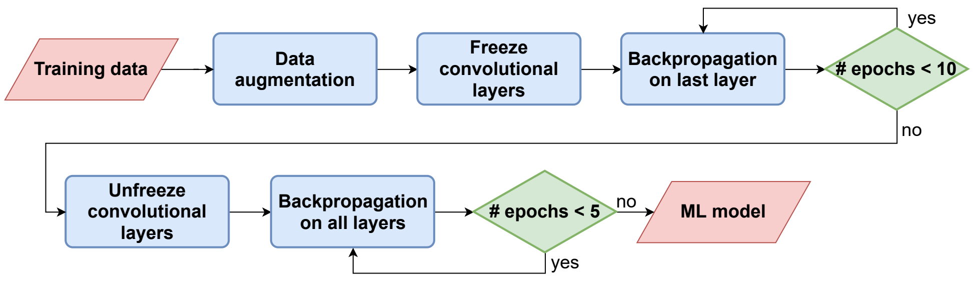


Source: the author.

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 I 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 I want to train CNN models for cell displacement and circuit wirelength, I 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.

4.3 CONVOLUTIONAL NEURAL NETWORK TRAINING PROCESS

Figure 10 – Flowchart of the proposed CNN training process.


Training data
Data augmentation
Freeze convolutional layers
Backpropagation on last layer
# epochs < 10
yes
no
Unfreeze convolutional layers
Backpropagation on all layers
# epochs < 5
no
yes
ML model

Source: the author.

In order to reduce the training time, I 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. I chose the SqueezeNet model (IANDOLA et al., 2016) as it presented the best results on our experimental evaluation (more details are given in Section 5.2.1). In addition, I applied the data augmentation method for training the model, as described in Chapter 2. The data augmentation process consists in applying transformations on the images generated by Algorithm 1 to artificially generate more data. This way, I can reduce over-fitting during the training process.

In Figure 10 the flowchart of the training process I used for the CNN models is shown. In order to decide on this flowchart I performed experiments with different training parameters. In the current section I am only going to mention the values that I chose for the parameters, but the process of deciding them will be detailed on Section 5.2.

The first step of the training process is data augmentation. I 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). I did not apply rotation and image distortions, as these transformations will generate images that are not realistic in the frame of circuit legalization. Figure 9 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, I freeze the weights of all convolutional layers and adjust only the weights of the last (fully connected) layer. This is done because, since the network was pretrained, at first I only want to adjust the weights on the fully connected layer, which is responsible for mapping the image features to the classes. This first step was executed for 10 epochs with a learning rate of 0.01. In the second stage, I 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 I 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 I only need to perform small changes on these weights. For this reason, I employ different learning rates for different sections of the neural network, as follows: I equally divide the neural network layers into sections, then I use learning rates ranging from 0.00001 to 0.002 on these 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 (HOWARD; RUDER, 2018). Details of how I chose the training parameters are provided in Section 5.2.2.

4.4 INTEGRATION WITH THE PHYSICAL DESIGN FLOW

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

The proposed CNN algorithm selector runs only one of the legalization algorithms, as described in Algorithm 5. CNN algorithm selector receives as input the placement P and an CNN model $\mathcal{M}$, which will select the best legalization algorithm $\mathcal{L}_{best} \in \mathcal{L}$ (line 1).

Algorithm 5: ALGORITHM_SELECTOR-CNN (P, $\mathcal{M}$)

```
   Input  : Placement $P$ and CNN model $\mathcal{M}$
   Output: Legalized circuit
1  $\mathcal{L}_{best} \leftarrow \mathcal{M}(P)$;
2  $\Lambda_{best}, result_{best} \leftarrow \mathcal{L}_{best}(P)$;
3  if $result_{best}$ then
4      foreach $c_i \in C$ do
5          $l(c_i) \leftarrow \lambda_{best}(c_i)$;
6      end
7  end
```

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 (CNN), I compare it to a reference version (LEG) that selects the best algorithm by actually executing all the legalization algorithms and comparing the results. The details of the implementation are given below.

The LEG selector algorithm was implemented for comparison purposes and runs all the legalization algorithms. In Algorithm 6, the implementation of the LEG version is given.

Algorithm 6: ALGORITHM_SELECTOR-LEG (P, $\mathcal{L}$)

```
Input  : Placement $P$ and list of legalization algorithms $\mathcal{L}$
Output: Legalized circuit
1  $\delta_{min} \leftarrow \infty$;
2  $\Lambda_{best} \leftarrow \emptyset$;
3  foreach $\mathcal{L}_i \in \mathcal{L}$ do
4      $\Lambda_i, result_i \leftarrow \mathcal{L}_i(P)$;
5      if $result_i$ then
6          $\delta_i \leftarrow$ EVALUATE_SOLUTION($\Lambda_i$, $P$);
7          if $\delta_i < \delta_{min}$ then
8              $\delta_{min} \leftarrow \delta_i$;
9              $\Lambda_{best} \leftarrow \Lambda_i$;
10         end
11     end
12 end
13 if $\Lambda_{best} \neq \emptyset$ then
14     foreach $c_i \in C$ do
15         $l(c_i) \leftarrow \Lambda_{best}(c_i)$;
16     end
17 end
```

The LEG selector receives as input a placement P 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). Observe that 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 locations found by the best solution, saved in Λ_{best} (lines 13–17).

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

It is important to observe that, while LEG needs to run all the legalization algorithms, CNN needs to run one algorithm, and the CNN model inference. As a consequence, the complexity of Algorithm 6 is proportional to the number of legalization algorithms available, while the complexity of Algorithm 5 is proportional in the worst case to the slowest legalization algorithm. Even though the CNN version of the algorithm selector still needs to run the CNN model, I have shown in Chapter 5 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 proportionally to the number of legalization algorithms to evaluate.

4.5 SUMMARY

This chapter presented the proposed algorithm selection framework for this thesis. First, I described how I generate the data to simulate three different stages of physical design as well as how the circuit snapshots were extracted. Then, I explained how the CNN model was trained and integrated into the physical design flow.

5 EXPERIMENTAL EVALUATION

5.1 EXPERIMENTAL SETUP

The experiments were performed on two Linux workstations for training and integrating the CNN model into the physical design flow. For training the proposed CNN model, I used 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. For the evaluation with the physical design flow, I used 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. I used the fast.ai 1.0.61 library (HOWARD; THOMAS, 2018) for training and the SqueezeNet architecture for transfer learning (IANDOLA et al., 2016). All experiments are available under public domain (EMBEDDED COMPUTING LAB, 2021) so they can be reproduced.

The model was trained and evaluated using ICCAD 2017 CAD Contest benchmarks (DARAV et al., 2017). 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, I can calculate how many sites are represented by each pixel[1] (shown in the rightmost column of the table).

Among the legalization algorithms available in the literature, **HAO** (LI et al., 2018), **ZIR** (ZHU et al., 2018) and **ODP** (from OpenDP) (DO; WOO; KANG, 2019) are the only academic ones adapted to the ICCAD 2017 benchmark and are able to handle the challenges present in modern circuits, such as multi-row cells. 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 the most appropriate 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 (DO; WOO, 2020).

I evaluated the proposed framework for the three cell movement strategies described in Chapter 4. Therefore, in the current chapter I present a section for each strategy denoting them as: cell movement simulating global placement (CMGP), cell movement simulating detailed placement (CMDP) and cell movement simulating incremental timing-driven placement (CMITDP). In order to investigate if the proposed framework is robust

[1] I used images with size of 500x500 pixels, as stated in Section 5.2. 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

enough to be useful for different physical design metrics, for each cell movement strategy I trained the models to classify the data using two different metrics. The first model selects the legalization algorithm that results in the lowest **cell displacement** for a given placement, whereas the second model 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*.

5.2 CNN SETUP

When using a ML model, it is important to select the training and validation parameters to achieve a better accuracy. In this Section I evaluate different parameters and their impact on the CNN quality. All these parameters were evaluated only for CMGP, and then the same parameters were used for the others scenarios.

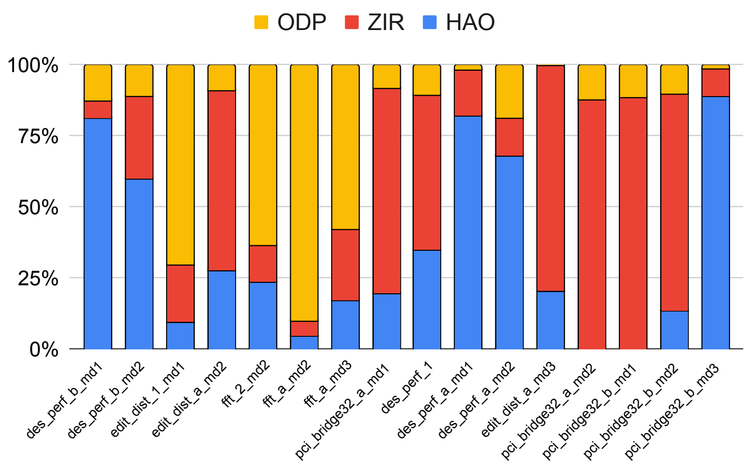
To increase the number of available legalization samples, I generate random illegal placements for each circuit to increase the amount of input data. In our experiments, for each benchmark circuit, one thousand placements were generated[2], resulting in a total of 16 thousand placements. The number of generated placements is a compromise between the CNN accuracy and the effort to generate data for it. If more placements were used, I expect that the CNN quality would improve, but since I observed good results using 1000 placements (as it will be seen later in this chapter), I did not generate more placements. I used 13 circuits for training and the remaining 3 for validation, resulting in about 20%

[2] Each placement was generated by applying random movements as described in Section 4 on the initial placement benchmarks provided by the ICCAD 2017 CAD Contest.

of the data being used for validation. I 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, I used a different validation set for each model. For Disp-CNN I used $des\_perf\_b\_md2$, $edit\_dist\_a\_md2$ and fft_2_md2 as the validation set. For WL-CNN I used $edit\_dist\_1\_md1$, fft_a_md3 and $des\_perf\_a\_md1$ as the validation set[3]. By separating the circuits this way I can verify if the CNN model can generalize its prediction to circuits that were not seen during training. Finally, I re-scaled all images to have the same size (500 $\times$ 500 pixels), even though I am using circuits of different sizes.

Figures 11 and 12 show the distribution of the legalization algorithms leading to the best results for each placement in terms of cell displacement and wirelength variation, respectively. These distributions were measured as for the CMGP case. 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.

Figure 11 – CMGP: distribution of best legalization algorithm in terms of cell displacement (y axis) for each placement generated from the benchmark set (x axis).

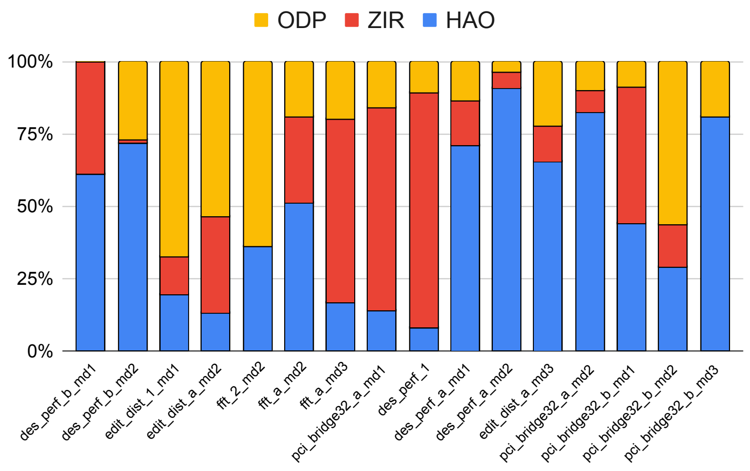


Source: the author.

I 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, I observe that they were at least 10% and 16% worse, respectively, than the best

[3] I also did experiments using 16-fold cross validation and observed very similar results. Therefore, this thesis reports only the results with a single validation set. The reader can refer to these results in our work published in Netto et al. (2021)

Figure 12 – CMGP: distribution of best legalization algorithm in terms of wirelength variation (y axis) for each placement generated from the benchmark set (x axis).

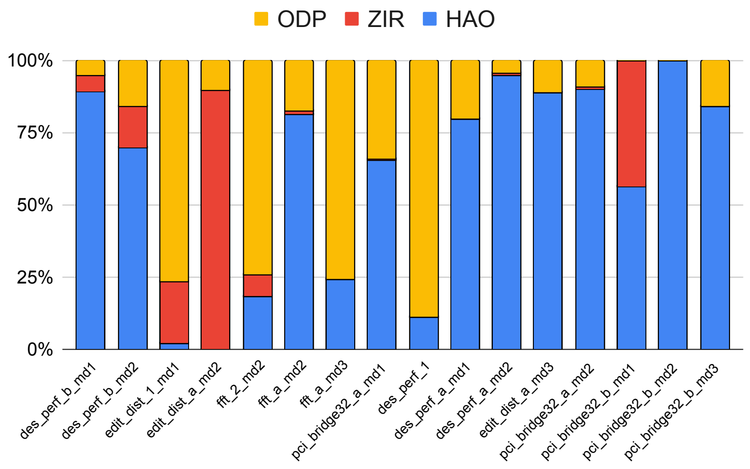


Source: the author.

solution in more than half of the cases. these results show that if I use only one of the three algorithms, I may lose in legalization quality.

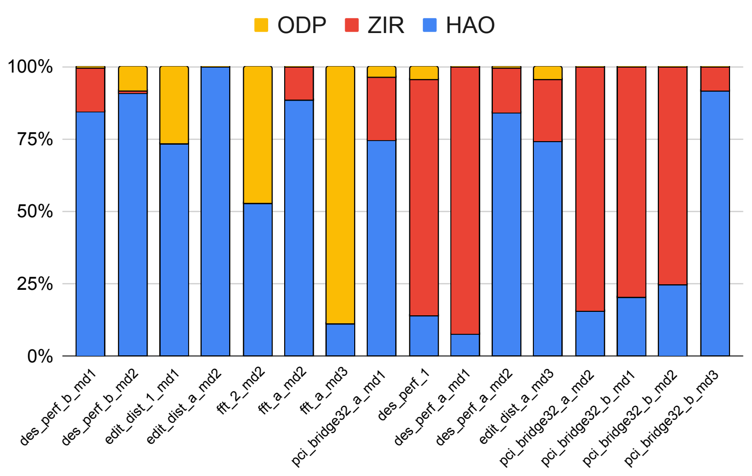
I repeated this experiment for the other two cell movement strategies. Figures 13 and 14 show the class distribution for CMDP, while Figures 15 and 16 show the distribution for CMITDP. It is possible to observe that for all three movement strategies there is no clear better algorithm. The samples where each legalization algorithm is the best is distributed among all circuits, and for each circuit there are cases where each of the three algorithms is the best.

Figure 13 – CMDP: distribution of best legalization algorithm in terms of cell displacement (y axis) for each placement generated from the benchmark set (x axis).



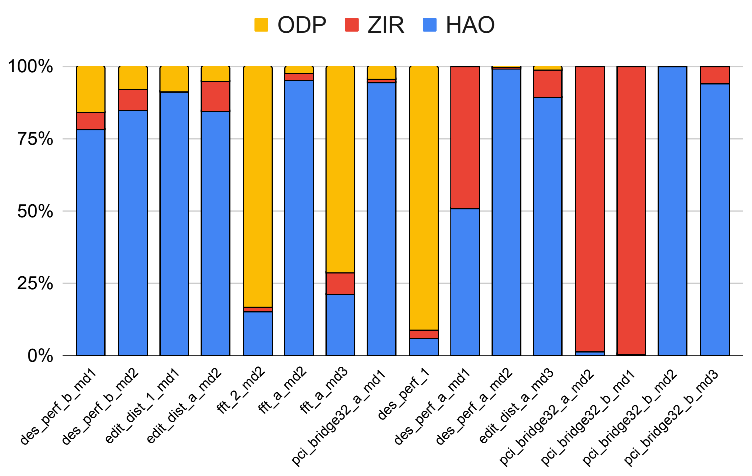
Source: the author.

Figure 14 – CMDP: distribution of best legalization algorithm in terms of wirelength variation (y axis) for each placement generated from the benchmark set (x axis).



Source: the author.

Figure 15 – CMITDP: distribution of best legalization algorithm in terms of cell displacement (y axis) for each placement generated from the benchmark set (x axis).



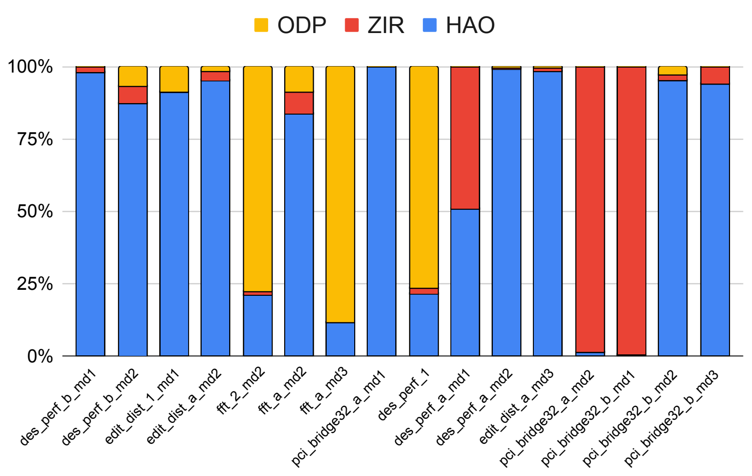
Source: the author.

5.2.1 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. I 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 of each CNN architecture (columns) for each class (rows). I focus on the WL-CNN model because it showed the lowest F-scores, making it the main model to optimize for this work's purpose.

The three architectures in Table 3 reach similar F-scores on average. However, AlexNet had the worst F-score in all the three cases, so it was discarded. When comparing

Figure 16 – CMITDP: distribution of best legalization algorithm in terms of wirelength variation (y axis) for each placement generated from the benchmark set (x axis).



Source: the author.

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

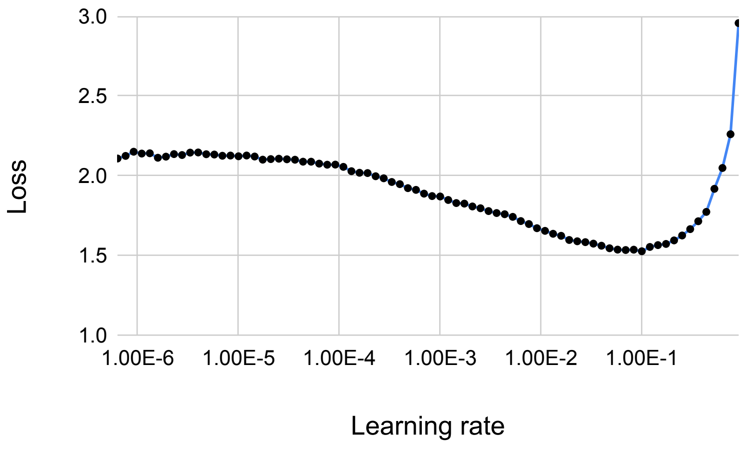
SqueezeNet to ResNet, the results are very similar, but SqueezeNet still achieved slightly better F-score for all classes. Therefore, I chose SqueezeNet for the experiments in this work.

5.2.2 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. I illustrate this phenomenon in Figure 17 where I show the training loss achieved for different learning rates executed for a few iterations of 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 I 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, I 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

Figure 17 – Training loss (x axis) for different learning rates (y axis) after a few iterations of training the WL-CNN model.

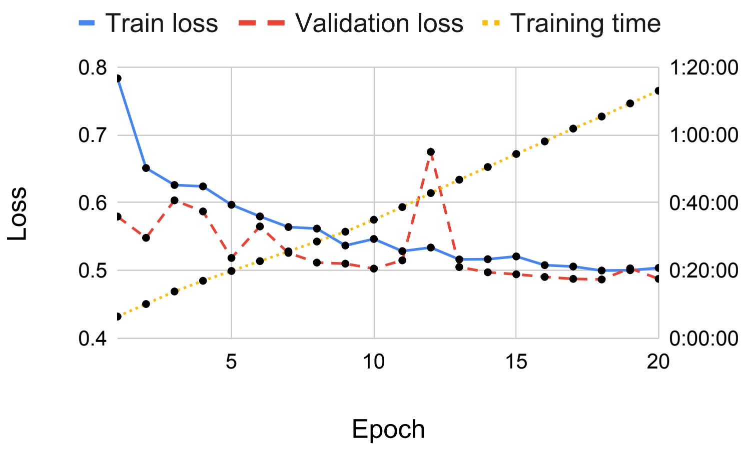


Source: the author.

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 I divided the training process in two stages (Section 4.3), I need to choose the number of epochs for each training stage.

In Figure 18, 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 much between epochs 10 and 20. Given this situation, I 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.

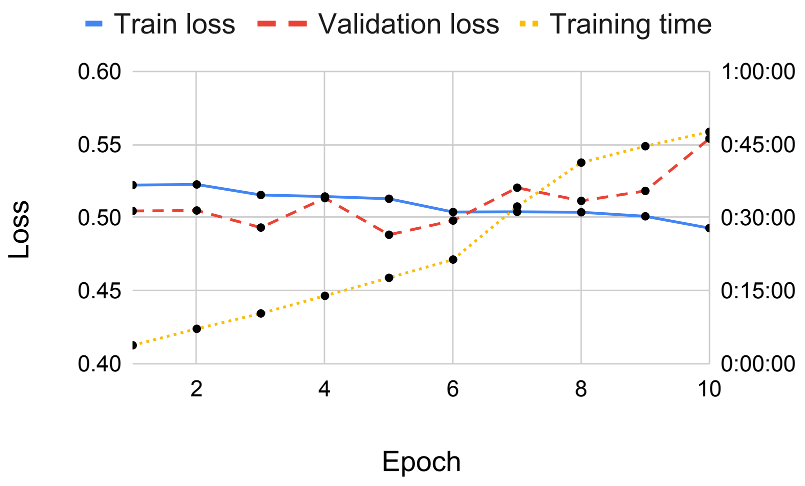
Figure 18 – Training loss, validation loss and training time (y axis) at each epoch (x axis) during the first stage of the training process of WL-CNN.



Source: the author.

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, I executed it only for 10 epochs. The measured training and validation losses, as well as training time are presented in Figure 19. 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 I may need more data to train this model for more epochs. As a consequence, I chose to run the second stage of training for 5 epochs only as to avoid over-fitting the network.

Figure 19 – 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.



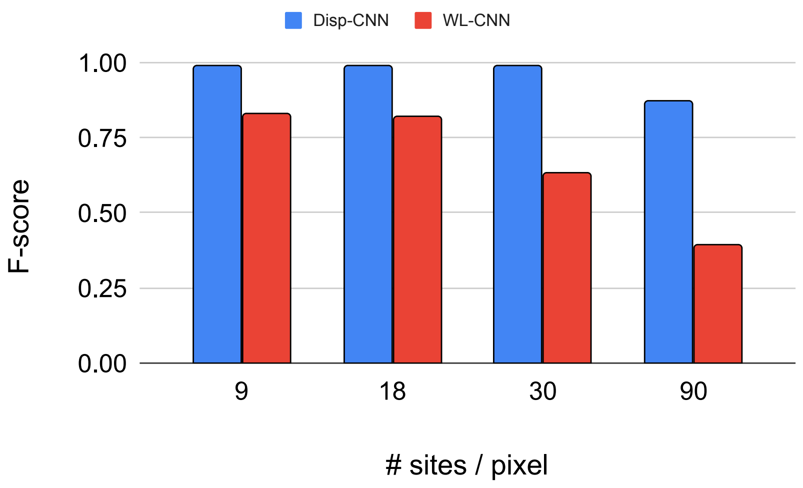
Source: the author.

5.2.3 CNN image size

To investigate how much the image size impacts in the quality of the CNN models, I 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 20 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).

I 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.

Figure 20 – F-score (y axis) of the CNN models for different image sizes (x axis).



Source: the author.

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

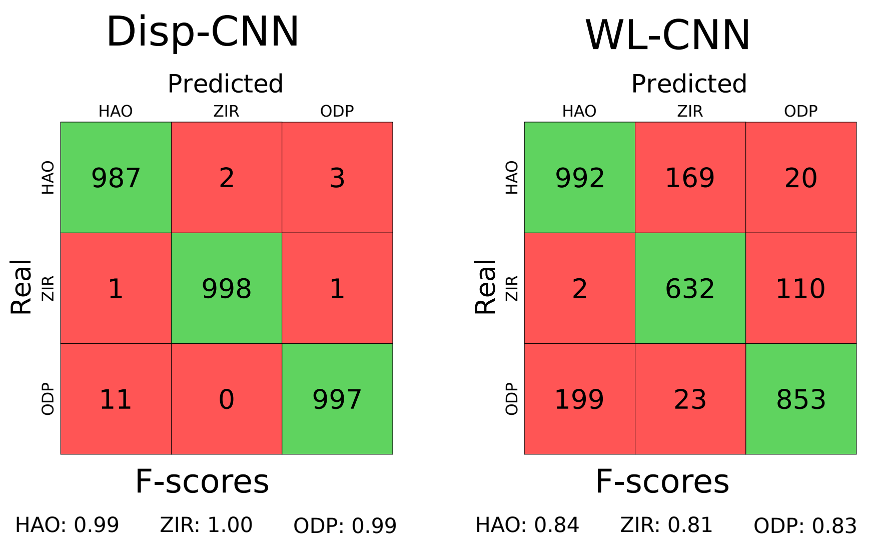
5.3 CELL MOVEMENT SIMULATING GLOBAL PLACEMENT

5.3.1 CMGP: Validation of the CNN models

The trained CNN models do not have a *true* class which I am looking for. Instead, there are three classes and I want the models to identify as accurately as possible instances of each class. Hence, I analyze the confusion matrices of both CNN models alongside their F-scores in Figure 21. 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 at least 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 40% of the instances for WL-CNN belong to the class with the highest F-score (HAO). Due to this

Figure 21 – Confusion matrices of CNN models for CMGP.



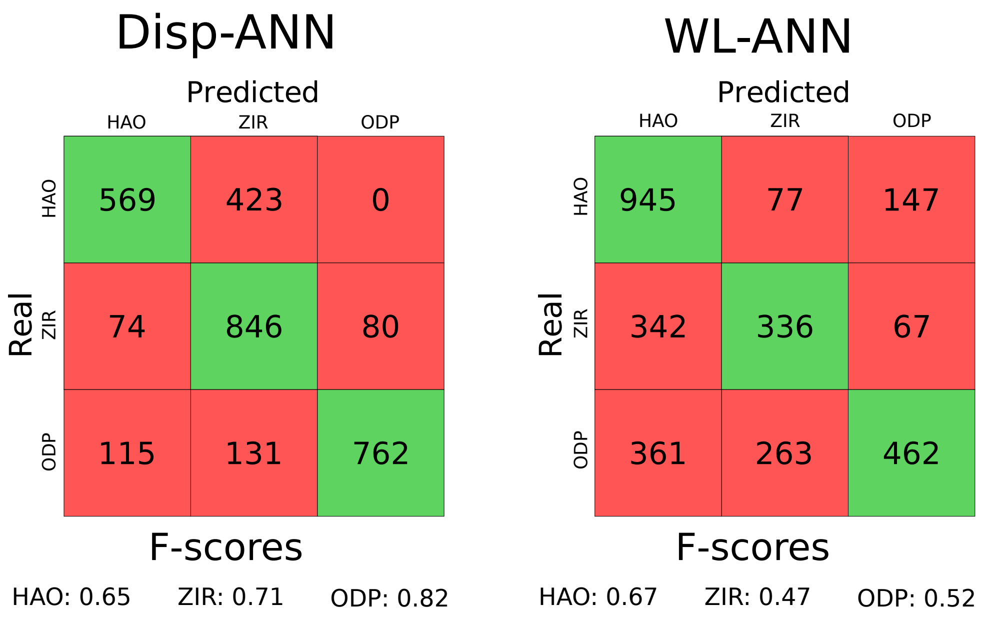
Source: the author.

imbalance, the model has more difficulty classifying instances of the other classes correctly. These results expose the ability of transfer learning to properly train deep learning 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 I am able to successfully train a deep CNN model with only 12.8k instances (see Section 5.1), with many of the CNN weights remaining stable due to the transfer learning process. If I had not used pre-trained weights, I would need to train the network for more epochs to achieve low training and validation losses. This would require more 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.

5.3.2 CMGP: Comparison with a related non-convolutional model

This section presents a comparison with a related non-convolutional model from the state of the art (NETTO et al., 2019). 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, I trained artificial neural networks (ANNs) for legalization outcome prediction using the same features from our previous work (NETTO et al., 2019). For fairness, I used as input data the same placements that were used as training and validation sets for the CNN models, but used the feature extraction strategy from (NETTO et al., 2019) instead of using the placement images. For example, the non-convolutional features include information such as the number of cells of different sizes in the design. In addition, I trained the ANNs for 15 epochs (the same total number of epochs as the CNNs). I trained models for both cell displacement and wirelength variation, and I denote them as Disp-ANN and WL-ANN, respectively.

Figure 22 – Confusion matrices of ANN models when using cell displacement and wirelength variation as evaluation metrics.



Source: the author.

The confusion matrices and F-scores of both ANN models are presented in Figure 22. When comparing them to the equivalent results of the Disp-CNN and WL-CNN (Figure 21 in Section 5.3.1), 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 5.2.1, 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.

Given that I observed that the ANN models are worse than the CNNs to predict the legalization algorithm to use, for all the other experiments in this thesis I am only going to evaluate CNNs. The reader can refer to the comparison of CNN and ANN in the physical design flow is in our work published on Netto et al. (2021).

5.3.3 CMGP: Evaluation with the physical design flow

After training the Disp-CNN and WL-CNN models, I integrated them into the physical design flow as described in Figure 8. For their evaluation, I generated five new random placements for each model, which ensures that the placements used in this integration step with the physical design flow were not previously used when training or validating the models. I used Algorithms 6 and 5 to implement two approaches for the algorithm selection, resulting in four variations of legalization flow:

- 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.
- WL-LEG: for each placement, runs the three legalization algorithms and selects the one with best wirelength improvement.
- WL-CNN: for each placement, uses the WL-CNN model to select only one legalization algorithm to execute.

Table 4 – Results of the algorithm selection framework when using cell displacement as the evaluation metric. Each row shows the cell displacement of one of the benchmarks. Each column specifies the algorithm that was used to legalize the circuit. Best results are highlighted in bold.

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

Table 4 shows the normalized results of this evaluation when using cell displacement as evaluation metric. For each design, the table shows the cell displacement after legalization when only HAO, ZIR or ODP is used, as well as, when Disp-LEG or Disp-CNN is used to select the legalization algorithm. All the results are normalized with respect to Disp-LEG, as this encompasses the best result for each circuit. It is worth remarking that in this table each row brings the sum of the results of five random placements for a benchmark circuit.

The first observation is that for every algorithm (HAO, ZIR and ODP), there is at least one case where it performs significantly worse than the best algorithm. For instance, HAO is the best algorithm for six designs, 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 designs 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 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 5.3.1.

Table 5 shows the results of evaluation with the physical design flow when using the wirelength variation as evaluation metric[4]. For each design, the results are normalized with respect to WL-LEG, as it always provides the best result. We can notice in the table that there are some designs where none of the direct algorithms (HAO, ZIR, and ODP) achieves a score of 1.00 (e.g., *des_perf_b_md*1 and *des_perf_a_md*1). This comes from the fact that each score is computed over five new random placements for each design, and the best legalization algorithm can change from one random placement to the other.

We can observe in Table 5 that HAO achieves a better average and median scores for the wirelength metric, which was not the case for cell displacement. However, if I were to use only HAO to legalize all the designs, I would still loose on legalization quality for some designs (such as *edit_dist_*1*_md*1 and *fft_a_md*3), as HAO delivers the best solution for only 5 out of 16 designs.

Although ZIR and ODP performed the best for some designs, 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_md*1 and *des_perf_a_md*2. 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 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 designs, 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_md*2 are very similar among the three legalization algorithms,

[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.

Table 5 – Results of the algorithm selection framework when using wirelength variation as the evaluation metric. Each row shows the wirelength variation of one of the benchmarks. Each column specifies the algorithm that was used to legalize the circuit. Best results are highlighted in bold.

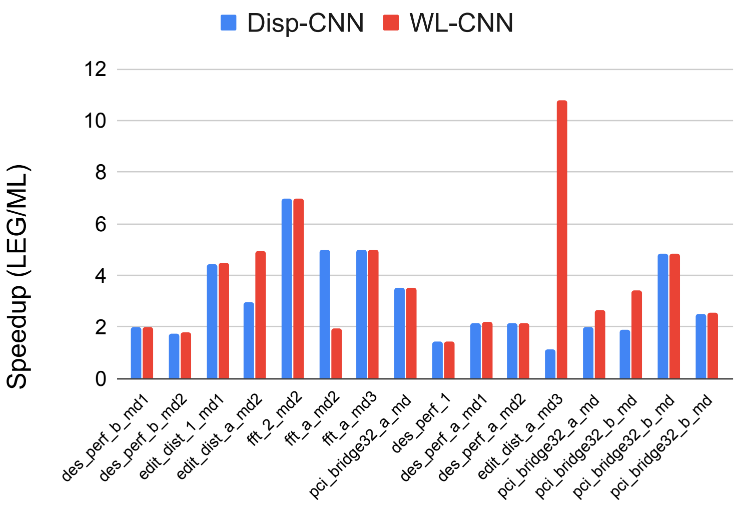
Design	Normalized wirelength variation				
	HAO	ZIR	ODP	WL-LEG	WL-CNN
des_perf_b_md1	1.03	1.04	1.77	**1.00**	1.03
des_perf_b_md2	**1.00**	1.23	1.03	**1.00**	**1.00**
edit_dist_1_md1	1.08	1.11	**1.00**	**1.00**	**1.00**
edit_dist_a_md2	1.03	1.01	1.02	**1.00**	1.02
fft_2_md2	1.01	1.10	**1.00**	**1.00**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

which makes it harder for the model to identify the best option during execution. For *fft_a_md*2, I 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 design that Disp-CNN classified incorrectly. This indicates that the CNN model is having some difficulty extracting the necessary features to predict for this design. 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.

5.3.4 CMGP: Speedup of the proposed CNN model

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 23 and Table 6. The figure illustrates the speedup as the average execution time ratio of LEG and CNN algorithms for each metric (cell displacement and wirelength) and each design. A speedup greater than 1 means that the CNN approach is faster than the baseline (LEG). The table presents the normalized runtime of each algorithm, as well as each algorithm selector (Disp-LEG, Disp-CNN, WL-LEG, WL-CNN). The runtimes and speedups were measured using a sample size of 10 executions.

Figure 23 – Speedup when using the algorithm selection framework (y axis) for each design (x axis). Disp-CNN results are shown in blue, and WL-CNN results are shown in red.



Source: the author.

Table 6 – Runtime of each legalization algorithm and the algorithm selector.

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

The variation in speedup among the designs is very noticeable in Figure 23, with some designs 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 as it can be observed in Table 6. In some situations, the best algorithm to use is also the slowest, which constrains the performance gain. Meanwhile, in other situations, the best algorithm is the fastest. This also explains the difference in speedup between Disp-CNN and WL-CNN for some designs, as the models can choose different algorithms for the different metrics. A notable case is *edit_dist_a_md*3, where the best algorithm for cell displacement was mostly ZIR, which is more than ten times slower than the other two algorithms for this design, resulting in a small speedup for Disp-CNN. On the other hand, the best algorithm for wirelength was mostly HAO, resulting in the large speedup of WL-CNN.

I 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 ranges from 25% where the best algorithm is fast for the design (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.

Another important observation is that the speedup achieved by using the framework scales with the number of legalization algorithms. For example, if I were to double the number of legalization algorithms considered (and the new algorithms had roughly the same total execution times as the algorithms considered), I 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, it would be possible to reduce the execution time of the LEG versions by running the legalization algorithms in parallel. However, there still would be speedup on the CNN side because the runtime can change a lot between the algorithms. For example, if one algorithm is 5x slower than the other two, the runtime even in the parallel version would be dominated by the slowest algorithm. If the best algorithm is significantly faster, then the CNN version will still achieve a good speedup.

5.4 CELL MOVEMENT SIMULATING DETAILED PLACEMENT

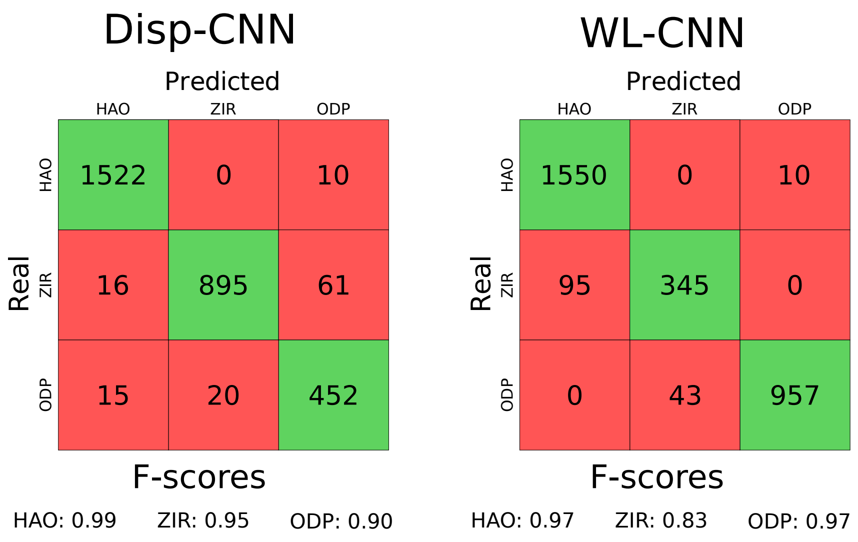
This Section shows the experiments performed using CMDP. I validated the CNN and evaluated their quality in the physical design flow. I used the same setup as described on Section 5.2.

5.4.1 CMDP: Validation of the CNN models

Figure 24 shows the confusion matrices of both displacement and wirelength models for CMDP, as described in Algorithm 3. Since I want to simulate detailed placement, I generated features after moving around 10% of the cells from a legal placement. This makes the placement illegal and, therefore, it is necessary to legalize it again.

The first thing to observe is that the data became much more unbalanced when compared to CMGP. Now, most of the samples are classified as HAO, and only few of them should be legalized with ZIR. Because of that, I selected the training and validation sets in a way that all classes are proportionally represented in both sets. The Disp-CNN model achieves an F-score of at least 0.90 for all classes. When analyzing the WL-CNN model, the F-score of ZIR degrades a little bit, due to a lower number of positive samples.

Figure 24 – Confusion matrices of CNN models for CMDP



Source: the author.

The F-scores of HAO and ODP are 0.97, while the F-score of ZIR is 0.83. However, I claim that this is still an acceptable F-score and since ZIR is the least represented class, it should not compromise the legalization quality in terms of displacement and wirelength, as it will be observed in the following results.

5.4.2 CMDP: Evaluation with the physical design flow

Table 7 shows the results of the integration of the CNN model into the physical design flow using the same approach as in CMDP. For each design, the table shows the cell displacement after legalization when only HAO, ZIR or ODP is used, as well as, when Disp-LEG or Disp-CNN is used to select the legalization algorithm. All the results are normalized with respect to Disp-LEG.

The first observation is that in some cases the algorithms failed to legalize the design. This happened for HAO in $edit\_dist\_1\_md1$ and for ODP in $pci\_bridge32\_a\_md1$ and $pci\_bridge32\_a\_md2$. Because of that, I do not report displacement results for them in the table. Also, it is important to highlight that the CNN model does not try to legalize these designs using an algorithm that fails. When analyzing the solution quality, HAO is better than the other two in almost all designs, as observed by the confusion matrices. However, there are still cases where HAO is significantly worse than the best algorithm, such as in fft_a_md3 and des_perf_1. ZIR is the best algorithm for only two designs, while ODP is the best one for 4 designs, including the one that HAO fails to legalize.

When analyzing Disp-CNN results, we can see that the framework achieved the same results as Disp-LEG for all designs except $pci\_bridge32\_b\_md1$. This design should be legalized with ZIR, and and the model fails to identify that. Even so, the result in this design was only 7% worse than Disp-LEG, and in average Disp-CNN achieves the same displacement that Disp-LEG. Finally, it is important to observe that Disp-CNN never

Table 7 – Results of the algorithm selection framework when using cell displacement as the evaluation metric. Each row shows the cell displacement of one of the benchmarks. Each column specifies the algorithm that was used to legalize the circuit. Best results are highlighted in bold.

Design	Normalized displacement				
	HAO	ZIR	ODP	Disp-LEG	Disp-CNN
des_perf_b_md1	**1.00**	1.14	1.62	**1.00**	**1.00**
des_perf_b_md2	**1.00**	1.11	1.51	**1.00**	**1.00**
edit_dist_1_md1	-	1.10	**1.00**	**1.00**	**1.00**
edit_dist_a_md2	1.03	**1.00**	1.44	**1.00**	**1.00**
fft_2_md2	1.10	1.39	**1.00**	**1.00**	**1.00**
fft_a_md2	**1.00**	1.06	1.13	**1.00**	**1.00**
fft_a_md3	1.26	1.34	**1.00**	**1.00**	**1.00**
pci_bridge32_a_md1	**1.00**	1.05	-	**1.00**	**1.00**
des_perf_1	1.20	1.09	**1.00**	**1.00**	**1.00**
des_perf_a_md1	**1.00**	1.13	1.38	**1.00**	**1.00**
des_perf_a_md2	**1.00**	1.17	1.38	**1.00**	**1.00**
edit_dist_a_md3	**1.00**	1.07	1.29	**1.00**	**1.00**
pci_bridge32_a_md2	**1.00**	1.05	-	**1.00**	**1.00**
pci_bridge32_b_md1	1.07	**1.00**	1.66	**1.00**	1.07
pci_bridge32_b_md2	**1.00**	1.09	1.21	**1.00**	**1.00**
pci_bridge32_b_md3	**1.00**	1.16	1.26	**1.00**	**1.00**
Average	1.03	1.12	1.12	**1.00**	**1.00**
Median	**1.00**	1.10	1.23	**1.00**	**1.00**

selected an algorithm that failed to legalize the design.

Table 8 shows the results of evaluation with the physical design flow when using the wirelength variation as evaluation metric. For each design, the results are normalized with respect to WL-LEG, as it always provides the best result. Similarly to the displacement scenario, HAO is the best algorithm in most cases, but it still fails for *edit_dist*_1_*md*1 and performs poorly for *des_perf*_1. Therefore, it is important that the CNN model can identify these cases.

Analyzing the WL-CNN results, it achieved the same wirelength as WL-LEG for all designs except *edit_dist_a_md*2, where it was 1% worse than WL-LEG. More importantly, it avoided HAO and ODP in the cases where they fail to legalize the design. In addition, the case where WL-CNN increased the wirelength by 1% was because HAO and ZIR have very similar results, so it failed to classify at least one of the five placements for this design. Therefore, we can observe that the CNN model is effective in predicting the best algorithm for CMDP.

5.4.3 CMDP: Speedup of the proposed CNN model

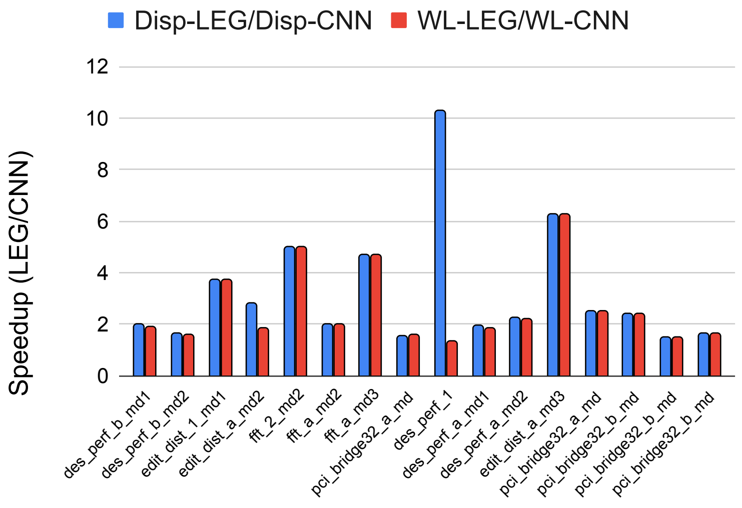
Figure 25 shows the speedup achieved by the CNN model in CMDP, while Table 9 shows the normalized runtime of each implementation (HAO, ZIR, ODP, Disp-LEG, Disp-CNN, WL-LEG, WL-CNN). The runtimes and speedups were measured using a sample size of 10 executions. Similarly to CMGP, the variation in speedup among the designs is very noticeable in Figure 25, with some designs having a speedup close to 2x, while the speedup of Disp-CNN for *des_perf*_1 is over 10. Once again, this variation comes from

Table 8 – Results of the algorithm selection framework when using wirelength variation as the evaluation metric. Each row shows the wirelength variation of one of the benchmarks. Each column specifies the algorithm that was used to legalize the circuit. Best results are highlighted in bold.

Design	Normalized wirelength variation				
	HAO	ZIR	ODP	WL-LEG	WL-CNN
des_perf_b_md1	**1.00**	1.08	1.60	**1.00**	**1.00**
des_perf_b_md2	**1.00**	1.19	1.89	**1.00**	**1.00**
edit_dist_1_md1	-	2.88	**1.00**	**1.00**	**1.00**
edit_dist_a_md2	1.01	1.04	2.77	**1.00**	1.01
fft_2_md2	1.02	1.27	**1.00**	**1.00**	**1.00**
fft_a_md2	**1.00**	1.07	1.22	**1.00**	**1.00**
fft_a_md3	1.01	1.04	**1.00**	**1.00**	**1.00**
pci_bridge32_a_md1	**1.00**	1.04	-	**1.00**	**1.00**
des_perf_1	1.09	**1.00**	1.14	**1.00**	**1.00**
des_perf_a_md1	**1.00**	1.12	1.44	**1.00**	**1.00**
des_perf_a_md2	**1.00**	1.47	1.91	**1.00**	**1.00**
edit_dist_a_md3	**1.00**	2.33	2.20	**1.00**	**1.00**
pci_bridge32_a_md2	**1.00**	1.38	-	**1.00**	**1.00**
pci_bridge32_b_md1	**1.00**	1.02	1.38	**1.00**	**1.00**
pci_bridge32_b_md2	**1.00**	1.06	1.09	**1.00**	**1.00**
pci_bridge32_b_md3	**1.00**	1.18	1.18	**1.00**	**1.00**
Average	1.05	1.32	1.55	**1.00**	**1.00**
Median	**1.00**	1.10	1.41	**1.00**	**1.00**

the different execution time of each possible legalization algorithm as it can be observed in Table 9. The best algorithm for *des_perf_1* was ODP, which was 10x faster than ZIR for this design. As consequence, avoiding to run all algorithms results in a significant speedup.

Figure 25 – Speedup when using the algorithm selection framework (y axis) for each design (x axis). Disp-CNN results are shown in blue, and WL-CNN results are shown in red.



Source: the author.

I also observed that the execution time of the CNN model's inference is still very small compared to the execution time of the legalization algorithms, even when fewer cells need to be legalized. This is because when the legalization algorithm moves the illegal cells to legal locations, it might need to move other cells to keep the whole design legal. As consequence, the execution time of the legalization is similar to the one from CMGP.

Table 9 – Runtime of each legalization algorithm and the algorithm selector normalized with respect to the best results.

Design	Runtime						
	HAO	ZIR	ODP	Disp-LEG	Disp-CNN	WL-LEG	WL-CNN
des_perf_b_md1	35	28	**25**	88	45	85	45
des_perf_b_md2	40	20	**20**	80	48	80	50
edit_dist_1_md1	43	30	**20**	93	25	93	25
edit_dist_a_md2	50	31	**20**	101	36	101	55
fft_2_md2	15	**5**	**5**	25	**5**	25	**5**
fft_a_md2	10	**5**	**5**	20	10	20	10
fft_a_md3	14	**5**	**5**	24	**5**	24	**5**
pci_bridge32_a_md1	10	5	**1**	15	10	15	10
des_perf_1	60	182	**15**	257	25	262	195
des_perf_a_md1	43	31	**20**	94	48	94	50
des_perf_a_md2	45	42	**25**	112	50	112	51
edit_dist_a_md3	55	300	**20**	375	60	375	60
pci_bridge32_a_md2	10	15	**0**	25	10	25	10
pci_bridge32_b_md1	15	11	**10**	36	15	36	15
pci_bridge32_b_md2	30	**5**	10	45	30	45	**30**
pci_bridge32_b_md3	30	**10**	**10**	50	30	50	30
Average	32	45	**13**	90	28	90	40
Median	33	18	**13**	65	28	65	30

5.5 CELL MOVEMENT SIMULATING ITDP

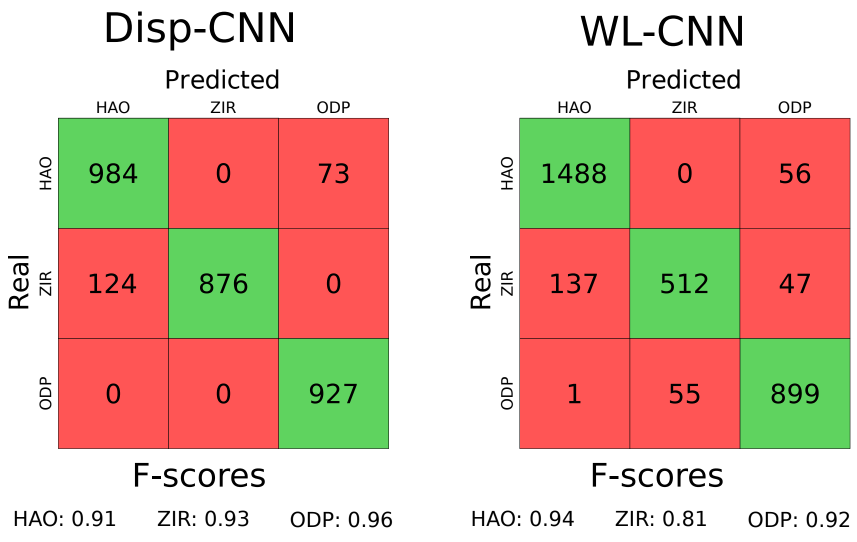
This Section shows the experiments performed using CMDP. I validated the CNN and evaluated their quality in the physical design flow. I used the same setup as described on Section 5.2. It is important to notice that the difference between this scenario and the previous one is that now the moved cells are not randomly selected, but some of them belong to the same circuit paths. The number of moved cells was about 15% on average per circuit.

5.5.1 CMITDP: Validation of the CNN models

Figure 26 shows the confusion matrices of both displacement and wirelength models for CMITDP. This is done by using Algorithm 4 to identify timing paths and moving cells in the same timing path. This makes the placement illegal and, therefore, it is necessary to legalize it again.

The first thing to observe is that the data is unbalanced as in CMDP, with most samples belonging to HAO class, and only a few of them being legalized with ZIR. Even so, the Disp-CNN model achieved good results in all the three classes, with an F-score of at least 0.91. When analyzing WL-CNN results, the model is still very good for both HAO and ODP, achieving an F-score of at least 0.92 in both of them, but worse for ZIR, where the F-score is 0.81. This is due to the lower number of samples for ZIR in the validation set. However, I still claim that since ZIR is the least represented class, these

Figure 26 – Confusion matrices of CNN models for CMITDP



Source: the author.

results should not compromise the wirelength of the solution, as it will be observed in the following results.

5.5.2 CMITDP: Evaluation with the physical design flow

Table 10 shows the results of the integration of the CNN model into the physical design flow using the same approach as in CMGP. For each design, the table shows the cell displacement after legalization when only HAO, ZIR or ODP is used, as well as, when Disp-LEG or Disp-CNN is used to select the legalization algorithm. All the results are normalized with respect to Disp-LEG.

The first observation is that some algorithms fail to legalize in the same cases as reported in CMDP. As consequence, I do not report the results for these cases, and it is important the CNN model avoid these algorithms when legalizing these designs. When analyzing the solution quality, the results are similar as in CMDP, as each algorithm is better in the same designs as it was for CMDP. However, the actual displacement results differ. For example, now HAO is very bad for des_perf_1, and this should be avoided by the CNN model. When analyzing Disp-CNN results, we can see that the framework achieved the same results as Disp-LEG for all designs. This correlates well with the confusion matrices, as the model achieved a high F-score for the three classes. This shows that the CNN model was very effective in identifying the correct algorithm in this scenario.

Table 11 shows the results of evaluation with the physical design flow when using the wirelength variation as evaluation metric. For each design, the results are normalized with respect to WL-LEG, as it always provides the best result. Similarly to the displacement scenario, HAO is the best algorithm in most cases, but it still fails for $edit\_dist\_1\_md1$ and has a large wirelength degradation for des_perf_1. On the other

Table 10 – Results of the algorithm selection framework when using cell displacement as the evaluation metric. Each row shows the cell displacement of one of the benchmarks. Each column specifies the algorithm that was used to legalize the circuit. Best results are highlighted in bold.

Design	Normalized displacement				
	HAO	ZIR	ODP	Disp-LEG	Disp-CNN
des_perf_b_md1	**1.00**	1.19	1.89	**1.00**	**1.00**
des_perf_b_md2	**1.00**	1.32	2.13	**1.00**	**1.00**
edit_dist_1_md1	-	1.20	**1.00**	**1.00**	**1.00**
edit_dist_a_md2	1.02	**1.00**	1.11	**1.00**	**1.00**
fft_2_md2	1.12	1.48	**1.00**	**1.00**	**1.00**
fft_a_md2	**1.00**	1.04	1.07	**1.00**	**1.00**
fft_a_md3	1.34	1.41	**1.00**	**1.00**	**1.00**
pci_bridge32_a_md1	**1.00**	1.06	-	**1.00**	**1.00**
des_perf_1	4.66	4.26	**1.00**	**1.00**	**1.00**
des_perf_a_md1	**1.00**	1.13	1.25	**1.00**	**1.00**
des_perf_a_md2	**1.00**	1.11	1.26	**1.00**	**1.00**
edit_dist_a_md3	**1.00**	1.03	1.14	**1.00**	**1.00**
pci_bridge32_a_md2	**1.00**	1.09	-	**1.00**	**1.00**
pci_bridge32_b_md1	1.03	**1.00**	1.77	**1.00**	**1.00**
pci_bridge32_b_md2	**1.00**	1.09	1.27	**1.00**	**1.00**
pci_bridge32_b_md3	**1.00**	1.24	1.41	**1.00**	**1.00**
Average	1.25	1.35	1.14	**1.00**	**1.00**
Median	**1.00**	1.12	1.13	**1.00**	**1.00**

hand, ZIR never achieves the best result for any design. However, it is still the best algorithm for a few placements. What happens is that it is never the best algorithm for all the five placements evaluated.

Table 11 – Results of the algorithm selection framework when using wirelength variation as the evaluation metric. Each row shows the wirelength variation of one of the benchmarks. Each column specifies the algorithm that was used to legalize the circuit. Best results are highlighted in bold.

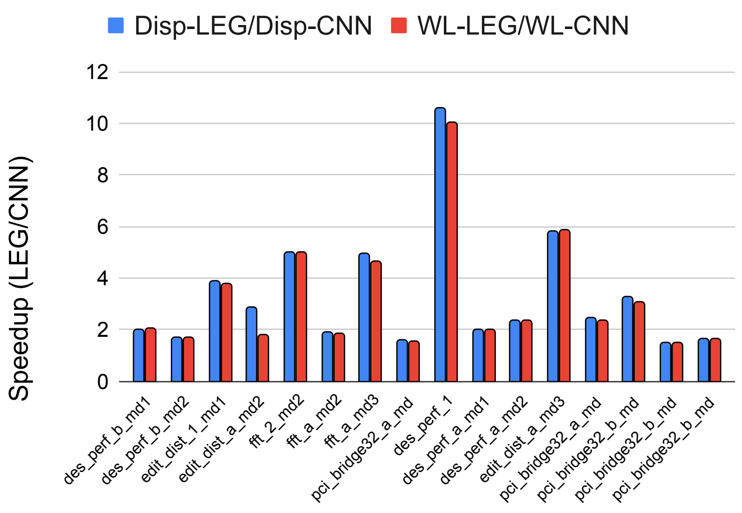
Design	Normalized wirelength variation				
	HAO	ZIR	ODP	WL-LEG	WL-CNN
des_perf_b_md1	**1.00**	1.07	1.58	**1.00**	**1.00**
des_perf_b_md2	**1.00**	1.16	1.91	**1.00**	**1.00**
edit_dist_1_md1	-	3.74	**1.00**	**1.00**	**1.00**
edit_dist_a_md2	**1.00**	1.58	2.64	**1.00**	**1.00**
fft_2_md2	1.02	1.32	**1.00**	**1.00**	**1.00**
fft_a_md2	**1.00**	1.07	1.20	**1.00**	**1.00**
fft_a_md3	1.04	1.09	**1.00**	**1.00**	**1.00**
pci_bridge32_a_md1	**1.00**	1.05	-	**1.00**	**1.00**
des_perf_1	3.57	3.16	**1.00**	**1.00**	**1.00**
des_perf_a_md1	**1.00**	1.17	1.28	**1.00**	**1.00**
des_perf_a_md2	**1.00**	1.41	1.90	**1.00**	**1.00**
edit_dist_a_md3	**1.00**	1.78	1.52	**1.00**	**1.00**
pci_bridge32_a_md2	**1.00**	1.62	-	**1.00**	**1.00**
pci_bridge32_b_md1	**1.00**	1.05	1.45	**1.00**	1.04
pci_bridge32_b_md2	**1.00**	1.05	1.08	**1.00**	**1.00**
pci_bridge32_b_md3	**1.00**	1.21	1.21	**1.00**	**1.00**
Average	1.24	1.53	1.49	**1.00**	**1.00**
Median	**1.00**	1.19	1.37	**1.00**	**1.00**

When targeting wirelength variation, the CNN model achieved the same wirelength as WL-LEG for all designs except *pci_bridge*32*_b_md*1, where it was 4% worse than WL-LEG. More importantly, it avoided HAO in the two cases where it is a bad algorithm to choose (*edit_dist_1_md1* and *des_perf_1*). As a result, the average wire-length variation of WL-CNN was the same than WL-LEG, while HAO is 24% worse on average. Therefore, we can observe that the CNN model is effective in predicting the best algorithm for CMITDP.

5.5.3 CMITDP: Speedup of the proposed CNN model

Figure 27 shows the speedup achieved by the CNN model for CMITDP, while Table 12 shows the normalized runtime of each implementation (HAO, ZIR, ODP, Disp-LEG, Disp-CNN, WL-LEG, WL-CNN). The runtimes and speedups were measured using a sample size of 10 executions. The speedup results are very similar to CMDP, with some designs having a speedup close to 2x, while the speedup of Disp-CNN for *des_perf_1* is over 10. Once again, this variation comes from the different execution time of each possible legalization algorithm as it can be observed in Table 9. However, one thing to notice is that there was not much variation between the speedup of the Disp and WL models. This happened because in this scenario the best algorithm for displacement was also the best one for wirelength in most cases. This can also be observed in Table 12, where the runtime of Disp-CNN and WL-CNN are similar for most designs.

Figure 27 – Speedup when using the algorithm selection framework (y axis) for each design (x axis). Disp-CNN results are shown in blue, and WL-CNN results are shown in red.



Source: the author.

As happened in CMDP, the execution time of the CNN model's inference is still very small compared to the execution time of the legalization algorithm, even though I am only moving cells in a few paths. The reason is the same as in the previous scenario: when the legalization algorithm moves the illegal cells to legal locations, it might need to move other cells to keep the whole design legal. As consequence, the CNN model remains effective in CMITDP.

Table 12 – Runtime of each legalization algorithm and the algorithm selector normalized with respect to the best results.

Design	Runtime						
	HAO	ZIR	ODP	Disp-LEG	Disp-CNN	WL-LEG	WL-CNN
des_perf_b_md1	31	26	**25**	82	41	81	40
des_perf_b_md2	35	22	**20**	77	45	75	44
edit_dist_1_md1	47	30	**20**	97	25	96	25
edit_dist_a_md2	60	35	**20**	115	40	113	63
fft_2_md2	15	**5**	**5**	25	**5**	25	**5**
fft_a_md2	12	**5**	**5**	22	12	21	12
fft_a_md3	15	**5**	**5**	25	**5**	23	**5**
pci_bridge32_a_md1	9	5	**1**	14	9	14	9
des_perf_1	55	195	**15**	265	25	251	25
des_perf_a_md1	44	35	**20**	99	49	97	49
des_perf_a_md2	45	45	**28**	118	50	118	50
edit_dist_a_md3	61	300	**20**	381	66	380	65
pci_bridge32_a_md2	10	15	**1**	25	10	24	10
pci_bridge32_b_md1	15	11	**10**	36	11	35	12
pci_bridge32_b_md2	30	**5**	10	45	30	45	30
pci_bridge32_b_md3	30	**10**	**10**	50	30	50	30
Average	32	47	**13**	92	28	90	30
Median	31	18	**13**	63	28	63	28

All these results demonstrate that the CNN models are accurate and effective predictors. They also show that the proposed algorithm selection framework is robust and can predict the best algorithm not only for cell displacement, but also for wirelength. This is a great advantage of using CNNs for prediction, as I 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.

5.6 SUMMARY

This chapter presented all the experiments performed in this thesis. It started by showing how I selected the parameters for training and evaluating the CNN model. Then, I evaluated the CNN models in the physical design flow for all the three cell movement scenarios. For each scenario, I measured the CNN model F-score, its impact on the cell displacement and wirelength variation during the physical design flow, and the speedup provided by the CNN model. The results presented in this chapter showed that the CNN-based algorithm selector as almost as good as running all the three legalization algorithms while being faster. In addition, the CNN never selected an algorithm that failed to legalize a placement, or that resulted in much worse results.

6 CONCLUSIONS

In this thesis I proposed a legalization algorithm selection framework that uses CNN to predict the best algorithm without having to run all the algorithm options. My hypothesis is that CNNs are well suited for this task since they are able to extract spatial properties from the data.

I extracted circuit snapshots as the data and the extensive experimentation on CNN training parameters showed how to select good parameters when using transfer learning for placement data. It is important to highlight that most works on ML for physical design do not provide this information, making them very hard to reproduce. This is specially important for neural networks, as these models have several parameters to be defined during training, and changing any of them can impact the final results. One specially important parameter is the image size. In this regards, I showed how big the image needs to be to represent circuits of a specific size. This shows that for larger circuits than those considered in this work, a partitioning strategy might be necessary to represent the data as images without losing too much information.

After selecting the CNN parameters, I presented an extensive experimental validation of the CNNs for the three cell movement scenarios I have explored on this thesis. For each cell movement scenario, I measured the F-score for the validation sets. The CNNs achieved an F-score of at least 0.81 for CMGP, at least 0.83 for CMDP and at least 0.81 for CMITDP, achieving F-scores close to 1.00 in a few cases. These results showed that the CNNs were able to accurately predict the correct legalization algorithm to use even in face of unbalanced data.

For the first cell movement scenario, I also compared the CNN results to a related ANN model. The ANN models performed significantly worse than the CNNs. For example, when using cell displacement as the target metric, the ANN had an F-score as low as 0.65 for one of the classes, when the CNN had F-scores of at least 0.99. When using wirelength variation as the target metric, the ANN performed even worse, with F-scores ranging form 0.47 to 0.52, while the CNN scored at least 0.81. These results confirm our claim that CNNs are better suited to this task as they are better able to extract spatial features from the data.

In the last set of experiments, I evaluated the trained CNNs when integrated into the physical design flow for the three cell movement scenarios. For most of the benchmark circuits, I observed that using the CNN models achieved exactly the same results when compared to running the three legalization algorithms. For the circuits that it did not achieved the same results, the degradation was at most 7%, showing that the CNNs can be used as an algorithm selector without compromising the flow quality. An important point to highlight is that in the conducted experiments the CNN never selected an algorithm that could not legalize the circuit.

Finally, I also observed that using the CNNs instead of running all three legalization algorithms was always faster. For example, the average runtime of the algorithm selector using CNN ranged from 28s to 46s, while the counterpart without using CNN ranged from 90s to 143s. For some circuits, the CNN offers a speedup of more than 10x to the algorithm selector. This was the case for circuit *edit_dist_a_md*3 in CMGP, where using the CNN reduced the runtime from 856s to 79s.

There are still a couple of points that can be investigated on future works. First, the same algorithm selection framework can be used with different evaluation metrics. This is easy to do by simply changing the feature extraction process to label the data according to the metric one wants to use. Similarly, it is also possible to test the framework with other benchmark sets, by generating data for them and retraining the CNNs.

A second investigation that can be done as future work is to experiment on legalization algorithms from industrial tools. Those algorithms are typically more powerful (and slower) than the academic ones, so it is not fair to compare academic and industrial tools. However, we can compare industrial tools from different vendors. This way, a designer with access to different tools could select the best one for a specific circuit.

Another possible future work relies on modeling algorithm selection as a regression problem. In this case, instead of predicting the algorithm to use as a label for the data, we need to predict the actual results of those algorithms, in order to compare them. This allows the designer to predict on different metrics and select the best algorithm based on a combination of those metrics. However, this is significantly harder to do as it is harder for the model to learn this.

Finally, a big challenge on the usage of ML for physical design, which remains open, is how to generalize the trained model for other technologies. In this thesis all circuits belong to the same technology node. Using a trained model on unseen circuits from the same technology node is important, but generalizing that to future technologies is significantly harder, as new nodes may add challenges that did not exist before. This is a challenge not only for this research, but for other works that use ML for physical design, and it is an incentive to keep working on this topic.

BIBLIOGRAPHY

ALPERT, C. J.; MEHTA, D. P.; SAPATNEKAR, S. S. **Handbook of algorithms for physical design automation**. Boca Raton, FL, United States: CRC press, 2008.

BARBOZA, E. C. et al. Machine learning-based pre-routing timing prediction with reduced pessimism. In: IEEE. **56th DESIGN AUTOMATION CONFERENCE (DAC)**. Piscataway, NJ, United States, 2019. p. 1–6.

BISHOP, C. M. **Pattern recognition and machine learning**. New York, NY, United States: Springer, 2006. 1–738 p.

CHAN, W.-T. J. et al. Routability optimization for industrial designs at sub-14nm process nodes using machine learning. In: ACM. **2017 INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN (ISPD)**. New York, NY, United States, 2017. p. 15–21.

CHEN, J. et al. Toward optimal legalization for mixed-cell-height circuit designs. In: ACM. **54th DESIGN AUTOMATION CONFERENCE (DAC)**. New York, NY, United States, 2017. p. 1–6.

CHOW, W.-K.; PUI, C.-W.; YOUNG, E. F. Legalization algorithm for multiple-row height standard cell design. In: IEEE. **53rd DESIGN AUTOMATION CONFERENCE (DAC)**. Piscataway, NJ, United States, 2016. p. 1–6.

DARAV, N. K. et al. Iccad-2017 cad contest in multi-deck standard cell legalization and benchmarks. In: **2017 INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN (ICCAD)**. Piscataway, NJ, United States: IEEE, 2017.

DO, S.; WOO, M. **OpenDP: Open Source Detailed Placement Engine**. 2020. [Online; accessed 12-October-2020]. Disponível em: https://github.com/The-OpenROAD-Project/OpenDP.

DO, S.; WOO, M.; KANG, S. Fence-region-aware mixed-height standard cell legalization. In: **2019 GREAT LAKES SYMPOSIUM ON VLSI (GLSVLSI)**. New York, NY, United States: ACM, 2019. p. 259–262.

EMBEDDED COMPUTING LAB. **Ophidian: an Open Source Library for Physical Design Research and Teaching**. 2021. [Online; accessed 23-November-2021]. Disponível em: https://gitlab.com/renan.o.netto/ophidian-research.

EVERITT, B. S. **The Cambridge dictionary of statistics**. Cambridge, England: Cambridge University Press, 2006. v. 106. 1–480 p.

GANDHI, U. et al. A reinforcement learning-based framework for solving physical design routing problem in the absence of large test sets. In: IEEE. **1st WORKSHOP ON MACHINE LEARNING FOR CAD (MLCAD)**. Piscataway, NJ, United States, 2019. p. 1–6.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. Cambridge, MA, United States: MIT press, 2016. http://www.deeplearningbook.org.

GUO, H.; HSU, W. H. **Algorithm selection for sorting and probabilistic inference: a machine learning-based approach**. 1–256 p. Tese (Doutorado) — Kansas State University, 2003.

HAN, S.-S. et al. A deep learning methodology to proliferate golden signoff timing. In: IEEE. **2014 DESIGN, AUTOMATION AND TEST ON EUROPE (DATE)**. Piscataway, NJ, United States, 2014. p. 1–6.

HE, K. et al. Deep residual learning for image recognition. In: IEEE. **2016 IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR)**. Piscataway, NJ, United States, 2016. p. 770–778.

HOWARD, J.; RUDER, S. Fine-tuned language models for text classification. **CoRR**, p. 1–12, 2018. Disponível em: http://arxiv.org/abs/1801.06146.

HOWARD, J.; THOMAS, R. **fast.ai - Making neural networks uncool again**. 2018. [Online; accessed 12-October-2020]. Disponível em: http://www.fast.ai/.

HUANG, G. et al. Densely connected convolutional networks. In: IEEE. **2017 IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR)**. Piscataway, NJ, United States, 2017. p. 4700–4708.

HUNG, C.-Y.; CHOU, P.-Y.; MAK, W.-K. Mixed-cell-height standard cell placement legalization. In: ACM. **2017 GREAT LAKES SYMPOSIUM ON VLSI (GLSVLSI)**. New York, NY, United States, 2017. p. 149–154.

HUNG, W.-T. et al. Transforming global routing report into drc violation map with convolutional neural network. In: ACM. **2020 INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN (ISPD)**. New York, NY, United States, 2020. p. 57–64.

IANDOLA, F. N. et al. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. **arXiv preprint**, p. 1–13, 2016.

IEEE. **International Technoogy Roadmap for Semiconductors**. 2015. [Online; accessed 02-December-2020]. Disponível em: http://www.itrs2.net/itrs-reports.html.

KAHNG, A. B. Machine learning applications in physical design: Recent results and directions. In: IEEE. **2018 INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN (ISPD)**. Piscataway, NJ, United States, 2018. p. 68–73.

KAHNG, A. B. et al. **VLSI physical design: from graph partitioning to timing closure**. New York, NY, United States: Springer Science & Business Media, 2011.

KAHNG, A. B.; LIN, B.; NATH, S. High-dimensional metamodeling for prediction of clock tree synthesis outcomes. In: IEEE. **2013 SYSTEM LEVEL INTERCONNECT PREDICTION WORKSHOP (SLIP)**. Piscataway, NJ, United States, 2013. p. 1–7.

KAHNG, A. B.; LUO, M.; NATH, S. Si for free: machine learning of interconnect coupling delay and transition effects. In: IEEE. **2015 SYSTEM LEVEL INTERCONNECT PREDICTION WORKSHOP (SLIP)**. Piscataway, NJ, United States, 2015. p. 1–8.

KAHNG, A. B.; MALLAPPA, U.; SAUL, L. Using machine learning to predict path-based slack from graph-based timing analysis. In: IEEE. **36th INTERNATIONAL CONFERENCE ON COMPUTER DESIGN (ICCD)**. Piscataway, NJ, United States, 2018. p. 603–612.

KAREN, S.; ZISSERMAN, A. Deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, 2014.

KERSCHKE, P.; TRAUTMANN, H. Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning. **Evolutionary computation**, MIT Press, Cambridge, MA, United States, v. 27, n. 1, p. 99–127, 2019.

KIM, M. et al. ICCAD-2015 CAD contest in incremental timing-driven placement and benchmark suite. In: **2015 INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN (ICCAD)**. Piscataway, NJ, United States: IEEE, 2015.

KOTTHOFF, L.; GENT, I. P.; MIGUEL, I. An evaluation of machine learning in algorithm selection for search problems. **AI Communications**, IOS Press, v. 25, n. 3, p. 257–270, 2012.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. **Advances in neural information processing systems**, v. 25, p. 1097–1105, 2012.

LI, H. et al. Routability-driven and fence-aware legalization for mixed-cell-height circuits. In: ACM. **55th DESIGN AUTOMATION CONFERENCE (DAC)**. New York, NY, United States, 2018. p. 1–6.

LIANG, R. et al. Drc hotspot prediction at sub-10nm process nodes using customized convolutional network. In: ACM. **2020 INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN (ISPD)**. New York, NY, United States, 2020. p. 135–142.

LIVRAMENTO, V. d. S. et al. Timing optimization during the physical synthesis of cell-based vlsi circuits. 2016.

NETTO, R. et al. How deep learning can drive physical synthesis towards more predictable legalization. In: **2019 INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN (ISPD)**. New York, NY, United States: ACM, 2019. p. 3–10.

NETTO, R. et al. Algorithm selection framework for legalization using deep convolutional neural networks and transfer learning. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, IEEE, 2021.

NETTO, R. O. et al. Aceleração da legalização incremental mediante o uso de árvores espaciais. 2017.

RUSSELL, S. J.; NORVIG, P. **Artificial intelligence: a modern approach**. New York, NY, United States: Pearson Education Limited,, 2016. 1–932 p.

SASAKI, Y. The truth of the f-measure. **Teach Tutor mater**, v. 1, n. 5, p. 1–5, 2007.

SMITH-MILES, K. A. Cross-disciplinary perspectives on meta-learning for algorithm selection. **CSUR**, ACM, v. 41, n. 1, p. 1–25, 2009.

STANFORD VISION LAB. **ImageNet**. 2016. [Online; accessed 4-June-2019]. Disponível em: http://www.image-net.org/.

TABRIZI, A. F. et al. A machine learning framework to identify detailed routing short violations from a placed netlist. In: IEEE. **55th DESIGN AUTOMATION CONFERENCE (DAC)**. Piscataway, NJ, United States, 2018. p. 1–6.

WANG, C.-H. et al. An effective legalization algorithm for mixed-cell-height standard cells. In: IEEE. **22nd ASIA AND SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE (ASP-DAC)**. Piscataway, NJ, United States, 2017. p. 450–455.

XIE, Z. et al. Routenet: routability prediction for mixed-size designs using convolutional neural network. In: IEEE. **2018 INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN (ICCAD)**. Piscataway, NJ, United States, 2018. p. 1–8.

YU, T.-C. et al. Pin accessibility prediction and optimization with deep learning-based pin pattern recognition. In: IEEE. **56th DESIGN AUTOMATION CONFERENCE (DAC)**. Piscataway, NJ, United States, 2019. p. 1–6.

YU, T.-C. et al. Lookahead placement optimization with cell library-based pin accessibility prediction via active learning. In: ACM. **2020 INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN (ISPD)**. New York, NY, United States, 2020. p. 65–72.

ZHOU, Q. et al. An accurate detailed routing routability prediction model in placement. In: IEEE. **6th ASIA SYMPOSIUM ON QUALITY ELECTRONIC DESIGN (ASQED)**. Piscataway, NJ, United States, 2015. p. 119–122.

ZHOU, Z. et al. Congestion-aware global routing using deep convolutional generative adversarial networks. In: IEEE. **1st WORKSHOP ON MACHINE LEARNING FOR CAD (MLCAD)**. Piscataway, NJ, United States, 2019. p. 1–6.

ZHU, Z. et al. Mixed-cell-height legalization considering technology and region constraints. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, IEEE, v. 39, n. 12, p. 5128–5141, 2020.

ZHU, Z. et al. Mixed-cell-height legalization considering technology and region constraints. In: IEEE. **2018 INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN (ICCAD)**. Piscataway, NJ, United States, 2018. p. 1–8.

APPENDIX A – LIST OF PUBLICATIONS AND AWARDS

A.1 PUBLICATIONS AS FIRST AUTHOR

Most of the results presented in this text were published as a journal paper in the IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD). Some earlier experiments with ANNs for physical design were published on the International Symposium on Physical Design (ISPD). The details of those publications are as follows:

A.1.1 IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)

- **Title:** Algorithm Selection Framework for Legalization Using Deep Convolutional Neural Networks and Transfer Learning (Published as early access on May 11, 2021)
- **Authors:** NETTO, Renan; FABRE, Sheiny; FONTANA, Tiago Augusto; LIVRAMENTO, Vinícius; PILLA, Laércio L.; BEHJAT, Laleh; GÜNTZEL, José Luís
- **DOI:** https://www.doi.org/10.1109/TCAD.2021.3079126

A.1.2 ACM International Symposium on Physical Design (ISPD 2019)

- **Title:** How deep learning can drive physical synthesis towards more predictable legalization
- **Authors:** NETTO, Renan; FABRE, Sheiny; FONTANA, Tiago; LIVRAMENTO, Vinícius; PILLA, Laércio; GÜNTZEL, José Luís
- **DOI:** http://dx.doi.org/10.1145/3299902.3309754
- **Best paper candidate:** https://ispd.cc/ispd2022/slides/ispd2019.html

A.1.3 Workshop on Open-source EDA Technology (WOSET 2018)

- **Title:** Ophidian: an Open-Source Library for Physical Design Research and Teaching
- **Authors:** NETTO, Renan; FONTANA, Tiago; FABRE, Sheiny; FERRARI, Bernardo; BARBATO, Thiago; LIVRAMENTO, Vinícius; GUTH, Chrystian; PILLA, Laércio; GÜNTZEL, José Luís

A.2 CONTRIBUTION TO OTHER PUBLICATIONS

During the doctorate, I also contributed to other publications, which although are not directly related to this thesis, they provided tools that were useful in the development of this thesis. Therefore, their details are provided below:

A.2.1 Design, Automation & Test in Europe (DATE2022)

- **Title:** CR&P: An Efficient Co-operation between Routing and Placement
- **Authors:** AGHAEEKIASARAEE, Erfan; TABRIZI, Aysa Fakheri; FONTANA, Tiago Augusto; NETTO, Renan; FABRE ALMEIDA, Sheiny; GANDHI, Upma; GÜNTZEL, José Luís; WESTWICK, David; BEHJAT, Laleh

A.2.2 IEEE Computer Society Annual Symposium on VLSI (ISVLSI 2021)

- **Title:** Integer Linear Programming-Based Global Routing With Cell Movement
- **Authors:** FONTANA, Tiago Augusto; AGHAEEKIASARAEE , Erfan; NETTO, Renan; FABRE, Sheiny; GANDHI, UPMA; TABRIZI, Aysa Fakheri; WESTWICK, David; GÜNTZEL, José Luís; BEHJAT, Laleh
- **DOI:** http://dx.doi.org/10.1109/ISVLSI51109.2021.00016

A.2.3 Symposium on Integrated Circuits and Systems Design (SBCCI 2018)

- **Title:** Enhancing Multi-Threaded Legalization Through k-d Tree Circuit Partitioning
- **Authors:** FABRE, Sheiny; GÜNTZEL, José Luís; PILLA, Laércio; NETTO, Renan; FONTANA, Tiago; LIVRAMENTO, Vinícius
- **DOI:** http://dx.doi.org/10.1109/SBCCI.2018.8533264

A.2.4 Symposium on Integrated Circuits and Systems Design 30 (SBCCI 2017)

- **Title:** Exploiting Cache Locality to Speedup Register Clustering
- **Authors:** FONTANA, Tiago; ALMEIDA, Sheiny; NETTO, Renan; LIVRAMENTO, Vinícius; GUTH, Chrystian; ALMEIDA, Sheiny; PILLA, Laércio; GÜNTZEL, José Luís
- **DOI:** http://dx.doi.org/10.1145/3109984.3110005

A.2.5 ACM International Symposium on Physical Design (ISPD 2017)

- **Title:** How Game Engines Can Inspire EDA Tools Development: A use case for an open-source physical design library
- **Authors:** FONTANA, Tiago; NETTO, Renan; LIVRAMENTO, Vinícius; GUTH, Chrystian; ALMEIDA, Sheiny; PILLA, Laércio; GÜNTZEL, José Luís

- **DOI:** http://dx.doi.org/10.1145/3036669.3038248

A.3 AWARDS

During the development of this thesis I also received the following awards:

- Best poster award on First IEEE CASS/CEDA Seasonal School on Electronic Design Automation (EDAS2020)
- Third place on the 2017 CAD Contest (Problem C: Multi-Deck Standard Cell Legalization)