TECHNISCHE UNIVERSITÄT DRESDEN

Elektrotechnik und Informationstechnik Institut für Feinwerktechnik und Elektronik-Design

Entwicklung eines maschinellen Lernverfahrens basierend auf neuronalen Netzen zur Vorhersage der Verdrahtbarkeit platzierter Schaltungslayouts

Matthias von Wachter

Geboren am: 25.10.1991 in München
Studienrichtung: Mikroelektronik

Studienarbeit

Betreuer
Dr.-Ing Robert Fischbach

Betreuender Hochschullehrer
Prof. Dr.-Ing. habil. Jens Lienig

Eingereicht am: 10. September 2019

Anstatt dieser Seite ist die originale, vom verantwortlichen Hochschullehrer unterzeichnete Aufgabenstellung einzubinden. Die weiteren abzugebenden Versionen der Diplomarbeit enthalten eine Kopie der Aufgabenstellung.
Das Binden der Studienarbeit hat so zu erfolgen, dass ein nachträglicher Seitenaustausch nicht möglich ist (keine Spiralbindung).

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig und ohne unzulässige Hilfe Dritter verfasst habe. Es wurden keine anderen als die in der Arbeit angegebenen Hilfsmittel und Quellen benutzt. Die wörtlichen und sinngemäß übernommenen Zitate habe ich als solche kenntlich gemacht. Es waren keine weiteren Personen an der geistigen Herstellung der vorliegenden Arbeit beteiligt. Mir ist bekannt, dass die Nichteinhaltung dieser Erklärung zum nachträglichen Entzug des Hochschulabschlusses führen kann.

Dresden, 10. September 2019

Matthias von Wachter

Kurzfassung

In modernen hochintegrierten Schaltungen mit Millionen bis Milliarden Transistoren wird es immer schwieriger auf dem Layout platzierte Bauelemente fehlerfrei im Detail zu verdrahten. Es ist also wünschenswert Probleme bei der Detailverdrahtung vorherzusagen ohne die rechenintensive Detailverdrahtung tatsächlich auszuführen. Diese Arbeit zeigt eine Verfahrensweise um das Vorkommen von Kurzschlüssen in der Detailverdrahtung anhand von aus einem platzierten Layout gewonnenen Daten vorherzusagen. Um für die Experimente notwendige Werkzeuge und Daten zu finden wurden ein Recherchen zu verfügbaren Layoutdaten und Werkzeugen für die Entwurfsautomatisierung mit einem besonderen Augenmerk auf Verdrahten und Platzieren betrieben. Ein neuronales Netzwerk wurde an die Charakteristika von aus dem platzierten Layout gewonnenen Daten angepasst. Die Vorhersagefähigkeiten dieses neuronalen Netzwerks waren zufriedenstellend bei Layouts mit vielen problematischen Sektoren mit einer Empfindlichkeit über 90%, fand aber weniger als die Hälfte der Kurzschlüsse in weniger fehlerbehafteten Entwürfen.

Abstract

In modern VLSI circuits with millions to billions of transistors it is getting harder to properly route a design after placement circuit elements. Therefore it is desirable to predict detailed routing problems without actually executing the computationally intensive detailed routing. This paper presents a method to predict the occurence of shorts in detailed routing with data extracted from a placed layout. Research into available layout data and tools for electronic design automation, with an emphasis on placement and routing, was done to get suitable tools and data for experimentation. A neural network was adapted to the characteristics of the extracted data. The prediction performance of the neural network was satisfactory for designs with a lot of problematic sectors with a sensitivity above 90% but found less than half of the shorts in less defect prone layouts.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Moderne hochintegrierte Schaltkreise (*Very Large Scale Integrated circuits* VLSI) bestehen aus zehntausenden bis hin zu Milliarden Transistoren. Ein rein manueller Entwurf ist in solchen Größenordnungen undenkbar. Daher werden Softwarewerkzeuge zur Entwurfsautomatisierung (EDA) eingesetzt.

Einer der letzten Schritte des damit ausgeführten Entwurfsablaufs, die Feinverdrahtung, verschlingt den Großteil der benötigten Rechenressourcen. Mit fortgeschrittenen Prozesstechnologien in der Halbleiterherstellung und damit einhergehender Komplexität der Entwurfsregeln (*design rules*) wird die Gewährleistung der Verdrahtbarkeit eines platzierten Layouts immer schwieriger.

Parallel zur steigenden verfügbaren Rechenleistung hat sich in den letzten Jahren das Feld des maschinellen Lernens stark entwickelt. Besonders im Bereich der Mustererkennung konnten große Erfolge erzielt werden. Es bietet sich an dies einzusetzen, um Muster in Layouts mit platzierten Bauelementen zu finden, die zu Problemen bei der Verdrahtung führen können.

Daher untersucht diese Arbeit inwiefern sich mit Verfahren des maschinellen Lernens Vorhersagen über die Verdrahtbarkeit eines platzierten Layouts treffen lassen. Kriterium dafür ist in dieser Arbeit die Existenz und Anzahl von Quadranten mit Kurzschlüssen des fertig verdrahteten Layouts.

1.1 Einführung in den Entwurfsablauf

Logiksynthese

Synthesewerkzeuge generieren ausgehend von einer Hardware-Beschreibung, verfasst in einer Sprache wie *Verilog* oder *VHDL*, eine Netzliste mit allen notwendigen Komponenten, zum Beispiel Standardzellen und Transistoren, sowie deren Verbindungen untereinander.

Partitionierung und Floorplanning

Bei der Partitionierung werden die Bauteile so in Gruppen geordnet, dass die meisten Verbindungen innerhalb dieser Gruppen sind und damit die Zahl der Verbindungen zwischen diesen Partitionen minimiert wird. Sinnvoll ist es auch Schaltungselemente, zwischen denen nur eine geringe Signalverzögerung (Latenz) existieren soll, in einer Gruppe zu sammeln. Der Floorplan ist eine räumliche Vorgabe, welche Partitionen in welchem Bereich des Chips platziert werden sollen. Mit ihm wird auch die absolute Größe und Form des Chips sowie Ort und Art der externen Anschlüsse festgelegt.

Platzierung

Die Platzierung legt die Orte aller Schaltungselemente auf der Chipfläche fest. Dabei müssen Randbedingungen wie das Vermeiden gegenseitiger Überlappungen beachtet werden. Optimiert wird unter Anderem die insgesamte Verbindungslänge und der kritische Pfad.[40, S.91] Der kritische Pfad ist die längste Verbindung zwischen zwei Elementen, hat somit die längste Latenz, und bestimmt dadurch die maximal mögliche Taktfrequenz des gesamten Chips.

Verdrahtung

Die Verdrahtung gliedert sich in zwei Phasen.

Die Globalverdrahtung ermittelt welche Gruppen von Bauelementen welche Gebiete zur Verbindung untereinander benötigen. Dazu wird die Chipfläche in Gitterzellen, im Englischen *grid cells* oder *gcells* genannt, unterteilt. Pro Gitterzelle prüft der Globalverdrahter wie die darin enthaltenen Schaltungselemente mit den Elementen in den anderen Gitterzellen verbunden werden sollen. Sogenannte Verdrahtungskanäle (*routing channels*), die die einzelnen Gitterzellen verbinden, werden bestimmt. Jede dieser Trassen hat eine bestimmte Kapazität an durch ihn passenden Leitungen. Für alle Verbindungen legt der Verdrahter dann fest durch welche Verdrahtungskanäle sie geführt werden.

Die Feinverdrahtung verbindet nun die einzelnen Bauteile miteinander im Rahmen der in der Globalverdrahtung bestimmten Trassen. Ein Netz verbindet zwei oder mehrere Anschlusskontakte von Bauelementen miteinander. Netze die rein innerhalb einer Gitterzelle liegen heißen lokale Netze, globale Netze stellen Verbindungen zwischen Gitterzellen her. Die Berechnung der Feinverdrahtungm benötigt viel Prozessorzeit.

Fertigungszentrierter Entwurf

Mit modernen Prozesstechnologien wächst die Herausforderung Schaltkreise mit immer kleineren Strukturbreiten zuverlässig mit nur wenig Ausschuss zu fertigen. Deshalb wird der fertigungszentrierte Entwurf, auch *Design For Manufacturability* (DFM) genannt, immer wichtiger. Zum Beispiel wird nach der Feinverdrahtung eine Layoutanpassung zur Verzerrungskorrektur bei Strukturen unterhalb der Lichtwellenlänge (*Optical Proximity Correction* OPC) durchgeführt.

1.2 Einführung in das Maschinelle Lernen

In letzter Zeit werden Verfahren des maschinellen Lernens auf eine Vielzahl von Problemen angewendet. Sie bieten sich besonders an um Muster in wenig strukturierten, großen Datenmengen zu erkennen. Beispiele sind die Bilderkennung [57], die Analyse von Emotionen in Text [32] und die Aufdeckung von Betrug in der Buchhaltung [55].

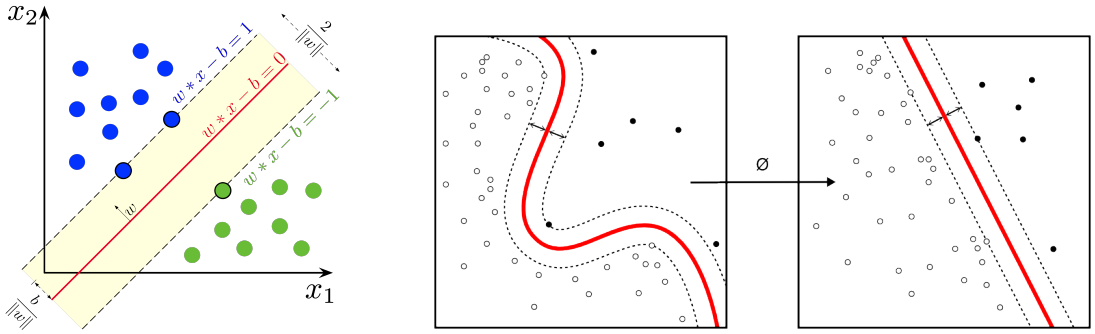
Maschinelles Lernen kann überwacht (*supervised*) oder unüberwacht (*unsupervised*) sein. In überwachten Verfahren lernt das Modell eine Beziehung zwischen Eingangsdaten und dafür erwünschten Ausgangsdaten, den Etiketten (*labels*). Unüberwachte Modelle finden Muster in Eingangsdaten ohne solche Etiketten, eignen sich aber nur für bestimmte Anwendungsfälle. Überwachte Modelle haben momentan mit Abstand die meisten praktischen Anwendungen. Daher konzentriert sich diese Arbeit auf überwachte Verfahren des maschinellen Lernens.

1.2.1 Stützvektormethode

Die Stützvektormethode, im Englischen *Support Vector Machines*(SVM) genannt, wird zur Klassifikation von Daten eingesetzt. Mit der Veröffentlichung des Buchs *The Nature of Statistical Learning* [65] von Vladimir Vapnik im Jahr 1995 gewannen SVM an Popularität. In den 90er

und 2000er Jahren waren sie den damaligen neuronalen Netzen in der Mustererkennung überlegen. Ein wichtiger Grund dafür ist, dass sie weniger Rechenressourcen benötigen als vergleichbare neuronale Netze.

Die Aufgabe eine Reihe an Werten in zwei Kategorien einzuteilen kann, wie in Abbildung 1.1a gezeigt, für den 2D Fall, als Menge an Punkten in einem n-dimensionalen Raum beschrieben werden. Diese sollen durch eine Hyperebene, also eine lineare Funktion, in zwei Gruppen der jeweils 'richtigen' Kategorie geteilt werden. Um auch neue Werte richtig klassifizieren zu können soll diese Hyperebene den größtmöglichen Abstand zu den Werten jeder Kategorie haben. Die Hyperebene wird deshalb mithilfe der zur ihr am nächsten gelegenen Werte definiert.



(a) Hyperebene $\vec{w}$ mit maximalem Abstand b zu den Trainingswerten [38] CC BY SA

(b) Kernel Maschine [4] CC 0

Abbildung 1.1: Support Vector Machine (SVM) (Stützvektormethode)

Aus diesen Stützwerten (*support vectors*) folgt der Name Support Vector Machine. Der beiderseitige Abstand b zu den Stützvektoren wird Bias genannt. Die gelbe Fläche in Abbildung 1.1a zwischen den beiden entlang der Stützvektoren aufgespannten Hyperebenen ist der zu maximierende Abstand (*margin*). Falls die Werte linear separierbar sind, es also mindestens eine die Werte korrekt trennende ungekrümmte Hyperebene gibt, kann $\|\vec{w}\|$ minimiert werden

$$\|\vec{w}\| = min! \tag{1.1}$$

unter der Bedingung, dass

$$y_i(\vec{w} * \vec{x}_i - b) \geq 1 \forall i \in [1; n] \tag{1.2}$$

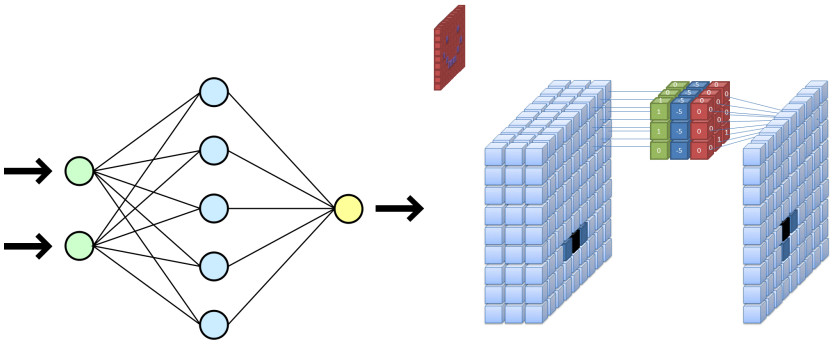
mit den Trainingswerten $\vec{x_i}$ und $y_i = 1$ für Werte oberhalb der trennenden Hyperebene und $y_i = -1$ für Werte unterhalb.

Falls die Trainingswerte $\vec{x}_i$ im gegeben endlich-dimensionalen Raum nicht linear separierbar sind, wie in Abbildung 1.1b zu sehen, können sie dennoch in einem höherdimensionalen Raum mit einem linearen Klassifikator getrennt werden. Die Transformation in diesen höherdimensionalen Raum ist aber sehr rechenintensiv. Effizienter ist es eine Kernelmethode anzuwenden Kernel sind mathematische Verfahren um einen Merkmalsraum (*feature space*) so zu transformieren, dass die Matrixprodukte implizit berechnet werden können ohne die aufwendige Transformation in den höherdimensionalen Raum explizit durchführen zu müssen. [37] gibt einen tieferen Einblick in das Forschungsfeld Kernelmethoden und deren Anwendung für SVM und andere Methoden des maschinellen Lernens.

1.2.2 Grundlagen Neuronaler Netze

Geschichte

Die Grundlagen neuronaler Netze werden seit den 1940er Jahren erforscht.[42] [25] [53] Die Modelle sollten ursprünglich Lernen in biologischen Systemen, wie dem menschlichen Gehirn, bestehend aus durch Synapsen verbundenen Neuronen erklären.



(a) Neuronales Netz mit Input Layer, Hidden Layer und Output Layer [16] CC BY

(b) Convolutional Neural Network mit Feature Layers [9] CC BY SA

Abbildung 1.2: Neuronales Netz

Ein zentraler Gedanke dabei ist, dass Synapsen nicht wie gewöhnliche Drähte eine konstante Leitfähigkeit haben, sondern sich diese während des Lernprozesses verändert. Diese veränderliche Leitfähigkeit wird das Gewicht (*weight*) einer Synapse genannt.

Es gab Bemühungen die Modelle auf Optimierungs- und Klassifizierungsprobleme anzuwenden. Zum Beispiel verwendete Meng-Lin Yu im Jahr 1989 Hopfield'sche neuronale Netze um die Platzierung von Bauelementen zu optimieren. Dies hatte jedoch nur bescheidenen Erfolg, sodass er im Fazit der Arbeit keine weitere Verwendung für Hopfield'sche neuronale Netze im Bereich der VLSI Entwurfsautomatisierung vorhersagte.[74] Anfang der 2010er Jahre war die verfügbare Rechenleistung so gestiegen, dass es möglich war bedeutend komplexere neuronale Netze zu simulieren. Mit einem Netzwerk bestehend aus 60 Millionen Gewichten und 500 000 Neuronen konnten Hinton et al. 2012 die bisher erreichte Genauigkeit in dem Bilderkennungswettbewerb *ImageNet* von 74% auf 86% steigern.[36] Seitdem dominieren neuronale Netze das Feld der Bilderkennung.[12, S.17]

Künstliche neuronale Netze

Beim überwachten maschinelles Lernen Verfahren stellen neuronale Netze eine Beziehung zwischen Eingangsdaten (Trainingsdaten) und Ausgangsdaten (Etiketten) her. Ein solches neuronales Netz ist definiert durch eine Anzahl von Knoten ('Neuronen') und deren gewichteter Verbindung ('Synapsen') untereinander. Beim Training des Netzes werden die Gewichte so verändert, dass die Verlustfunktion minimiert wird.

Verlustfunktion

Eine Verlustfunktion (*loss function*) bildet ab, wie nah die vom Netzwerk vorhergesagten Daten an dem erwünschten Ergebnis sind. Ein gut trainiertes Netzwerk hat nur noch einen kleinen Verlust (*loss*).[12, S.11] Ein einfaches Beispiel ist die quadratische Verlustfunktion L mit einem vorhergesagten Wert x, einem Zielwert z und dem anpassbaren Parameter α

$$L(x) = \alpha(z - x)^2 \tag{1.3}$$

Damit die Optimierung des Netzwerks funktioniert muss die Verlustfunktion stetig differenzierbar sein.

Backpropagation

Mitte der 1980er Jahre wurde eine Möglichkeit gefunden, um neuronale Netze effektiv zu trainieren.[12, S.15] Bei der Fehlerrückführung (*backpropagation*) wird die Information, ob eine Vor-

hersage richtig oder falsch war, also Veränderungen im Ergebnis der Verlustfunktion, genutzt um die Gewichte, die für diese Vorhersage verantwortlich waren, so anzupassen, dass der Verlust sinkt.

1.2.3 Deep Learning und Convolutional Neural Networks

Charakteristisch für Deep Learning ist die Aufteilung des Netzwerks, siehe Abbildung 1.2a, in mehrere Schichten (*layer*). Der Vorteil von Deep Learning ist, dass komplexe Zusammenhänge, wie etwa in Bildern, Schicht (*layer*) für Schicht abstrahiert werden. Für viele praktische Anwendungen ist es außerdem wichtig, dass das Modell nicht alle Daten auf einmal lernen muss, sondern sie sich auch in einzelnen Bündeln (*batches*) aneignen kann. Erkenntnisse aus neu hinzugekommene Daten können somit in das Modell einfließen, ohne alle Daten von neuem zu verarbeiten (*online learning*). Solches *online learning* ist mit anderen Ansätzen für maschinelles Lernen, wie etwa der Stützvektormethode, nicht möglich.[12, S.23]

Die Eingangsdaten werden mit einem Wert pro Neuron in der Eingangsschicht (*input layer*) aufgenommen. Diese Daten werden dann in einer oder meist mehreren ausgeblendeten Schichten (*hidden layers*) verarbeitet.

Abbildung 1.2b veranschaulicht die Funktionsweise der ausgeblendeten Schichten in einem Convolutional Neural Network (CNN). Durch die mathematische Operation der Faltung (*convolution*) werden Zusammenhänge zwischen den einzelnen Eingangswerten, hier drei Pixeln eines Bildes, hergestellt. Dadurch werden Eigenschaften (*features*) benachbarter Gruppen von Pixeln extrahiert, um dann in der nächsten Schicht wiederum Zusammenhänge zwischen diesen Gruppen herzustellen.

So werden die Informationen komprimiert, zusammengefasst und zur Ausgangsschicht hin geführt. Diese Schicht besteht aus einem oder mehreren Knoten, bzw. Neuronen. Falls zum Beispiel entschieden werden soll, ob ein Gesicht in einem Eingangsbild lächelt oder nicht, kann der Wert eines Ausgangsneurons für die Wahrscheinlichkeit eines Lächeln stehen.

2 Stand der Technik

2.1 Vorhersage von Verdrahtungsfehlern

2.1.1 Globale Verdrahtung und Overflow

Der herkömmliche Ansatz zum Finden von problematischen Sektoren im platzierten Design ist die Globalverdrahtung. Sie betrachtet Gitterzellen, auch Gcells (*grid cells*) genannt. Das sind Gitterzellen in die das Design unterteilt wird. Sie ermittelt die notwendigen Verbindungen zwischen diesen Sektoren. Ein Resultat der Globalverdrahtung ist der Overflow, das Verhältnis von notwendigen Verbindungen zu verfügbarem Platz dafür an jeder Kante der Gitterzellen.

Total overflow (TOF)

Um diese Metrik relevanter zu machen addiert der *total overflow*(TOF) alle Verbindungselemente auf, welche die Kapazität an Gitterzellkanten übersteigen: [5]

$$TOF = \sum_{k} max(0, Bedarf(e) - Kapazitaet(k)) \tag{2.1}$$

Durchschnitt der schlechtesten X%

Um sich unter Vernachlässigung der einfach verdrahtbaren Bereiche auf die Schwachstellen zu konzentrieren, wird der Durchschnitt der problematischsten X% genommen. Sortiert werden die Kanten der Gcells entweder nach Schwere der Engpässe (*congestion*) oder die Netze nach den gravierendsten Engpässen einer sie schneidenden Kante. [5]

Gesamtlänge der Verbindungen (RWL)

Die insgesamte Drahtlänge (*total routed wirelength*(RWL) kann Aufschluss über den Aufwand (*effort*) geben, den der Globalverdrahter aufwenden muss. Die Untergrenze dafür ist die Länge des minimalen rektilinearen Steinerbaums, der alle Bauelemente verbindet. [40, S.142f] Sollte die RWL deutlich darüber liegen wäre das ein Zeichen, dass der Globalverdrahter viele lokale Engpässen umgehen musste, um Overflow zu vermeiden. [5]

Anzahl der Panoramanetze

Ein Aspekt, den die RWL nicht erfasst, ist die Notwendigkeit Verdrahtungshindernissen (*routing blockages*) auszuweichen. Netze, die einen großen Bogen um Hindernisse machen werden im Englischen *scenic routes* genannt. Das bezieht sich auf die längere Strecke, die Autofahrerinnen auf

sich nehmen, um ein Panorama (*scenic view*) zu genießen. Eine hohe Anzahl von Panoramanetzen weist neben dem erhöhten Verdrahtungsaufwand auch auf eine schlechtere Verzögerungscharakteristik (*timing*) aufgrund der längeren Drähte (*wires*) hin. [5]

Anzahl der Netze jenseits von X%

Die Anzahl der Netze, die Gitterzellen mit Engpässen jenseits von X% schneiden, verschafft einen guten Eindruck davon, wieviel Arbeit noch notwendig ist um den Entwurf fehlerfrei zu verdrahten. [5]

Anzahl der Fehlermeldungen

Dem Verdrahter kann vorgeschrieben werden wie streng er die bereits erwähnten Panorama Routen und andere unerwünschte Entwurfseigenschaften vermeiden soll. Solche Einschränkungen (*constraints*) resultieren wenn sie verletzt werden (*violation*) in Fehlermeldungen. Die Anzahl dieser Fehlermeldungen gibt einen Hinweis auf die noch nötige Arbeit, um den Entwurf konform mit allen Entwurfsregeln (*design rules*) fertig zustellen.

Limitierungen der Globalverdrahtung

Neue Technologien der Halbleiterfertigung mit sehr kleinen Strukturgrößen (*feature sizes*) setzen stetig komplizierter werdende Entwurfsregeln voraus damit die Schaltung bei guter Ausbeute (*yield*) fertigbar bleibt. Eine Herausforderung ist, dass ab dem 32nm Technologieknoten (*technology node*) die Metallebenen (*metal layer*) keine einheitliche Drahtbreite und Dicke mehr haben um je nach Signalart ungünstige Leitungseigenschaften auszugleichen. [5] Diese und andere Einschränkungen bedingen eine Vielzahl an Platzierungs- und Verdrahtungsblockaden (*placement and routing blockages*).

Für den Globalverdrahter wird es also immer schwieriger alle für die erfolgreiche Feinverdrahtung notwendigen Faktoren zu berücksichtigen. Kritisch sind auch rein lokale Netze, da sie die Grenzen der Gitterzelle nicht schneiden und daher nicht in die Statistiken zur Engpässen einfliessen.

Dadurch reichen die oben beschriebenen Indikatoren auf Basis der Globalverdrahtung bei neueren Technologieknoten nicht mehr aus um Verdrahtungsfehler vorherzusagen.

2.1.2 Verfahren des maschinellen Lernens

Um die komplexen Muster, die von der Globalverdrahtung nicht vollständig berücksichtigt werden können, zu erfassen, bieten sich Verfahren des Maschinellen Lernens an. In diesem Bereich gibt eine große Menge wissenschaftlicher Literatur, aber keine Patente speziell für die Suche nach Verdrahtungsfehlern.

An Accurate Detailed Routing Routability Prediction Model in Placement

Zhou et al. wenden diesen Ansatz auf den Datensatz aus dem Wettbewerb des *International Symposium on Physical Design* 2014 (ISPD2014) [48] an, um Probleme in der Feinverdrahtung vorherzusagen. [76] Sie passen die Größe einer Gitterzelle so an, dass etwa 70 * 70 Gitterzellen einen Entwurf abdecken.

Die Eingangsdaten für das Modell (*input features*) werden sowohl aus dem platzierten Design als auch einer schnellen Globalverdrahtung gewonnen.

Die Dichte der Anschlusskontakte (*pins*) und Ort von Verdrahtungsblockaden kann nach der Platzierung der Bauelemente ausgelesen werden. Unter einer gewissen Schwelle machen mehr

oder weniger geringe Anschlussdichten keinen Unterschied für die Verdrahtbarkeit. Nur sehr hohe Dichten behindern die Verdrahtung. Deshalb werden nur Werte jenseits von 60% der durchschnittlichen Dichte verwendet und der Rest auf Null gesetzt. Eng angeordnete Anschlüsse können Probleme verursachen, auch wenn die durchschnittliche Dichte gering ist. Die Erfassung der Anschlussverteilung hilft solche Probleme vorherzusagen. Dafür wurde jeder Anschluss um den Mindestabstand (*min spacing*) um ihn herum erweitert (*bloated*). Das Verhältnis, wie diese erweiterten Pins sich gegenseitig überlagern, (*coverage ratio*) ergab die Anschlussverteilung.

Die schnelle Globalverdrahtung bringt Informationen über lokale, semilokale und globale Netze. Da die meisten Anschlüsse laut den Autoren höher als breit sind, können horizontale Verbindungen einfacher verdrahtet werden als vertikale. Deswegen wurde der Overflow separat in horizontaler und vertikaler Richtung erfasst. Außerdem unterscheiden sie zwischen geraden Verbindungen und solchen über Eck, da sie beobachtet haben, dass gerade Verbindungen mehr Verdrahtungsressourcen benötigen als solche über Eck. Als Maß für das Vorkommen von lokalen Netzen in einer Gitterzelle verwendeten sie deren *half perimeter wirelength* (HPWL), also die Hälfte des Umfangs des die Netzpins umschließenden Rechtecks. Die Verletzungen von Entwurfsregelnwerden pro betroffenem Netz gezählt. Das hat den Vorteil, dass es mit dem Aufwand die Fehler zu beheben korreliert. Zudem verändert sich die Zahl der betroffenen Netze weniger sprunghaft als die absolute Zahl der Fehlermeldungen.

Neben den erwähnten Eingangsdatentypen testeten die Autoren noch weitere Parameter. Diese verschlechterten jedoch lediglich die Vorhersagekraft des Modells.

Die Autoren ermittelten Korrelationen zwischen diesen Eingangsdaten und der Anzahl der Kurzschlüsse im fertig feinverdrahteten Entwurf mit dem statistischen Verfahren multivariate adaptive Regression.

Je nach betrachtetem Entwurf bewegte sich die Vorhersagegenauigkeit zwischen 79% und 88%.

Routability Optimization for Industrial Designs at Sub-14nm Process Nodes Using Machine Learning

In ihrem 2017 erschienen Artikel [10] zeigen Chan et al. die, von Alpert et al. im Jahr 2010 beschriebenen und vorhergesagten, Schwächen der Ermittlung der Verdrahtbarkeit anhand von Ergebnissen der Globalverdrahtung (siehe Abschnitt 2.1.1) in einem modernen Design. Während Alpert et al. die Probleme der Globalverdrahtung für Entwürfe mit einer 32nm Prozesstechnologie beschrieben verwenden Chan et al. eine bedeutend neuere - und damit noch komplexere - 'sub 14nm' Technologie. Hier folgte aus einem Overflow in der Globalverdrahtung nur in 24% der Fälle eine tatsächliche Verletzung der Entwurfsregeln bei der Feinverdrahtung. Ihr auf maschinellem Lernen basierendes Modell erreicht mit 74% eine gut dreimal so hohe Genauigkeit.

In ihrer Arbeit nutzen sie dieses Modell nicht nur, wie Zhou et al. in [76], um Vorhersagen zu treffen, sondern verbessern damit die Verdrahtbarkeit nach der Platzierung.

Das Modell nutzt ähnliche Eingangsdaten wie [76]. Allerdings unterteilen sie jede Gitterzelle der Globalverdrahtung noch einmal in mehrere kleinere Quadranten, von den Autoren 'lokale Fenster' genannt, um die Ursachen von Problemen genauer erfassen zu können. Die Dichte der Anschlusskontakte und die der Entwurfszellen (*library cells*)werden ebenso erfasst wie Ergebnisse der Globalverdrahtung, zum Beispiel Overflow, Nachfrage und Angebot an Verdrahtungsressourcen jeder Metallschicht und Durchkontaktierungsschicht (*via layer*). Die Autoren ermittelten, inspiriert von Chan et al. in [66], den durchschnittlichen und kleinsten Abstand zwischen den Anschlusskontakten. Zellen die überdurchschnittlich häufig in Regionen mit Verletzungen von Entwurfsregeln auftratenwerden als 'unfreundlich' klassifiert. Ähnlich behandeltwerden ein Vielfaches einer Zeile (*row*) hohe und sequentielle angeordnete Zellen. Deren Eingangslasten (*fan in*), Ausgangslasten (*fan out*) und Häufigkeitwerden gespeichert. Die Eingangs- bzw. Ausgangslast bezieht sich auf die Anzahl und Kapazität mit einem Bauteil verbundener Bauteile. Verbindungsparameter beschrieben die Anzahl von Netzen mit ausschließlich lokalen Anschlusskontakten, die das lokale Fenster schneidenden Netze sowie die Zahl der verbundenen Anschlusskontakte jenseits

des Fensters.

Aufgrund der Schwierigkeit an Entwürfe für Technologieknoten unter 14nm zu Forschungszwecken zu kommen hatten die Autoren nur einen einzigen Entwurf um ihr Modell zu trainieren und zu testen. Sie kreierten unterschiedliche Layouts indem sie den gleichen Entwurf mit unterschiedlichen Einstellungen für die Optimierung und Platzierung synthetisierten. Sie nutzten 20% der Gitterzellen um ihre Modelle zu trainieren und die restlichen 80% um die trainierten Modelle zu testen. Die 20/80 Aufteilung wendeten sie 12 mal zufällig an und nahmen die Vorhersagegenauigkeit jedes Durchlaufs auf. Um den Effekt den umliegende Strukturen auf die Verdrahtbarkeit eines Qaudranten haben zu berücksichtigen, betrachteten sie sowohl kleine Gitterzellen als auch große, sich überlappende Gitterzellen in unterschiedlichen Größen. Das deutlich seltenere Auftreten von Entwurfsregelfehlern im Vergleich zu fehlerfreien Quadranten kompensierten die Autoren durch die höhere Gewichtung von DRC Fehlern im Lernprozess.

Mit diesen Daten probierten sie mehrere Methoden um das Vorhersagemodell zu trainieren aus. Dabei maßen sie jedesmal die statistische Signifikanz der einzelnen Datentypen und verglichen den Ort der vorhergesagten Entwurfsregelfehler mit den tatsächlich aufgetretenen Fehlern. Der Anteil der falsch-positiv Treffer wurde unter 0,5% gehalten während sie die richtig-positiv Rate iterativ verbesserten. Die erprobten mathematischen maschinelles Lernen Verfahren waren lineare Regression, logistische Regression und Support Vector Machines(SVM). Support Vector Machineshatten mehr Vorhersagekraft als die Regressionsverfahren. Die besten Ergebnissewerden dabei mit dem *Leaps* Paket der auf Datenvisualisierung spezialisierten Programmiersprache *R* erzielt. Es ermittelt automatisiert die Eingangsparameter mit der höchsten Korrelation zu den erwünschten Ausgangskenngrößen.

Mit Hilfe ihres Vorhersagemodells konnten die Autoren nun aktiv die Verdrahtbarkeit eines Designs verbessern. Dazu machten sie es sich zunutze, dass in den meisten praktisch angewendeten Entwurfsflows die Dichte der Standardzellen begrenzt wird. Es ist also in der Umgebung jeder Verletzung der Entwurfsregeln etwas freier Platz vorhanden. Allerdings ist dieser oft etwas weiter entfernt außerhalb der Gitterzelle und kann somit nicht automatisch zur Feinverdrahtung lokaler Netze herangezogen werden. Das Werkzeug der Forscher gab hier Abhilfe, indem es solchen ohnehin vorhandenen Freiraum in die Nähe von vorhergesagten Entwurfsfehlern verschob. Die Zahl der Entwurfsregelverletzungen sank im Durchschnitt um 20%.

A Machine Learning Framework to Identify Detailed Routing Short Violations from a Placed Netlist

2018 erreichen Tabrizi et al. auch ohne Nutzung von Daten aus der Globalverdrahtung eine hohe Genauigkeit von durchschnittlich 90% bei der Vorhersage von Kurzschlüssen (*Shorts*). Die Entwürfe auf die sie ihr Trainingsmodell anwendeten stammen aus dem Wettbewerb für akademische Werkzeuge zur Entwurfsautomatiserung ISPD 2015 [49].

Er greift die Entwürfe aus dem Vorjahr auf und erweitert diese um Grenzen für die Dichte der Logikzellen, Regeln über den Ort der platzierten Bauteile (*region constraints*), unterschiedliche Nutzungsgrade der Standardzellen (*standard cell utilizations*) und Platzierungs- und Verdrahtungsblockaden. [7]

Sie bearbeiten also die etwas komplexere und wirklichkeitsnähere Version der von Zhou et al. in [76] verwendeten Benchmarks. Die Entwürfe von 2014 und 2015 werden von den Organisatoren des Wettbewerbs als 'sub 45nm' beschrieben, also eine deutlich ältere Prozesstechnologie als die von Chan et al. verwendeten 'sub 14nm'.

Dabei beschränken sie sich auf Eingangsdaten aus dem platzierten Layout, ohne die Globalverdrahtung zu berücksichtigen.

Die Eingangsdaten des Modells (*features*) sind ähnlich den von Zhou et al. in [76] und Chan et al. in [66] verwendeten, mit der bereits erwähnten Ausnahme der Daten aus der Globalverdrahtung. Es wird zudem berücksichtigt, wie nah am Zentrum eine Gitterzelle ist, da sich dort viele globale Netze kreuzen und somit die Gefahr von Engpässen steigt. Die neu hinzugekom-

menen Platzierungs- und Verdrahtungsblockaden sowie Makros werden als Hindernisse für die Zugänglichkeit einer Gitterzelle für den Verdrahter in den unterschiedlichen Metallschichten berücksichtigt. Die Konzentration von Anschlussstellen wird nicht durch die Überschneidung ihrer erweiterten Flächen, sondern durch die Standardabweichung ihrer Positionen in horizontaler und vertikaler Richtung bestimmt. Außerdem wird der Effekt des Clock Netzwerks durch die Anzahl von Clock Pins in einem Quadranten erfasst. Wieviele Netze eine Gitterzelle in horizontaler und vertikaler passieren ist eine weitere Metrik. Verfügbarkeit von Verdrahtungspfaden (*tracks*) fand genauso Berücksichtigung wie die Existenz von Verbindungen zu Netzen die nach nicht standardmäßigen Regeln (*Non default routing net*/NDR) verdrahtet werden sollen. All diese Datenwerden auf ihre Standardabweichung hin normiert.

Die Autoren wählten ein neuronales Netz als maschinelles Lernen Modell. Das erwähnte Ungleichgewicht im Auftreten, bzw. Nichtauftreten, von Fehlern konnte das Modell das neuronale Netz selbst ausgleichen, da die Entwurfsregelverletzungen in der Verlustfunktion durch das Netz gewichtetwerden.

Das verwendete neuronale Netz hatte nur eine ausgeblendete Schicht (*hidden layer*) mit 20 Knoten.

Die Entwürfe aus dem ISPD 2015 Benchmarkwerden mit dem bei diesem und dem vorjährigen Wettbewerb erfolgreichen akademischen Platzierer Eh?Placer [43] platziert. Verdrahtet wurde es mit Mentor Graphics Olympus Router, dem gleichen Werkzeug mit dem auch die Layouts des ISPD 2014 und 2015 generiertwerden.

2.2 Vorhersage von Lithographie-Hotspots

Konventionelle Lithographie verwendet Lichtquellen mit einer Wellenlänge von 193nm um Halbleiterwafers zu belichten. Bei Strukturgrößen unter 193nm stößt diese Technologie an ihre Grenzen. Eine mehrfache, jeweils leicht versetzte Belichtung (*multiple patterning*) ermöglicht dennoch Strukturgrößen weit unter der verwendeten Lichtwellenlänge. Solche Kompensationsmethoden sind stark abhängig von den geometrischen Formen der zu fertigenden Schaltungen. Die Ausbeute kann durch einzelne Orte mit für die Lithographie ungünstiger Geometrie (Hotspots) sinken. Die herkömmliche Art solche Hotspots zu finden ist die physikalische Simulation des Lithografieprozesses für den gesamten Chip. Die dafür notwendigen Rechenressourcen sind aber gerade bei komplexeren Designs so hoch, dass andere Verfahren notwendig werden um ein schnelles Iterieren im Entwurfsprozess zu ermöglichen. Das Ziel ist also, im Gegensatz zur bisher behandelten Suche nach möglichen Hindernissen für die Verdrahtbarkeit, geometrische Muster zu finden die zu Problemen bei der Lithografie führen können.

2.2.1 Wissenschaftliche Arbeiten

Layout Hotspot Detection with Feature Tensor Generation and Deep Biased Learning

Yang et al. behandeln die Suche nach solchen Lithografie Hotspots. [73] Das verwendete Datenmaterial stammt aus dem ICCAD 2012 [64] Wettbewerb. Es umfasste ein für eine 32nm Technologie bestimmtes Layout und vier 28nm Layouts. Alle Datensätze sind fertig platzierte und verdrahtete Layouts mit Hinweisen auf den Ort von Lithographie-Hotspots. Wie gut die Leistung ihres Modells auf große Entwürfe skaliert testeten die Autoren mit einem aus allen 28nm Benchmarks zusammengesetzten Layout und drei nicht näher identifizierten kommerziellen Layouts. Gemein mit den ISPD Wettbewerben haben sie, dass die Zahl des gesuchten Defekts, Verdrahtungsfehler oder Lithografie Hotspot, sehr klein gegenüber der Zahl der untersuchten Strukturen ist. Dementsprechend müssen 'Treffer', also Hotspots, stärker gewichtet (*biased*) werden als unproblematische Sektoren.

Bei der Lichtbrechung spielt die räumliche Anordnung eine deutlich größere Rolle als bei der Verfügbarkeit von Verdrahtungsressourcen. Daher verarbeiten die Autoren Bilddaten und nicht

Statistiken über die jeweils betrachtete Gitterzelle. Jede Gitterzelle ist bei ihnen in noch kleinere Unterregionen aufgeteilt. Diese Unterregionen stellen sie mittels Diskreter Kosinustransformation (*Discrete Cosine Transform*, DCT) als endliche Summe von Frequenzkomponenten dar. Die Transformation hat den Vorteil, dass es die in den Bilddaten vorhandene Information unter Beibehaltung des lokalen Kontextes komprimiert, ähnlich wie die auch auf DCT basierende JPEG-Bildkompression. [47] Dadurch kann das mit Faltung arbeitende neuronale Netz (*Convolutional Neural Network*, CNN) weniger Daten und besser auf es angepasste Daten verarbeiten. Das Trainieren und Vorhersagen wird deutlich effizienter.

Multi Batch Gradient Descent (MGD) ermöglicht es das Modell mit neu hinzugefügten Daten anzulernen (*Online Learning*) ohne den kompletten Trainingsprozess wiederholen zu müssen. MGD ermöglicht es auch moderne hochparallele Rechensysteme, wie zum Beispiel Grafikkarten, besser auszunutzen als wenn alles in einer einzigen Charge (*batch*) auf einmal berechnet würde. Dadurch konnten die Experimente mit einer Nvidia K620 Grafikkarte gesteuert von einem Intel Xeon E5 Prozessor durchgeführt werden.

Die erzielte durchschnittliche Genauigkeit der neuronales Netz Architektur beträgt 95,5%, bei gleichzeitiger Verlängerung der Rechenzeit um maximal 50% zum nächstbesten Konkurrenten [75], ist laut den Autoren eine signifikante Verbesserung gegenüber der Literatur zum Zeitpunkt der Artikelveröffentlichung.

2.2.2 Patente

Im Bereich der Anwendung von maschinellem Lernen auf Probleme der Entwurfsautomatisierungwerden selbstverständlich auch Patente angemeldet. In Patenten soll technisches Wissen über neue Erfindungen offenbart werden, um im Gegenzug der Patentanmelderin für einen begrenzten Zeitraum ein Monopol auf die kommerzielle Anwendung der Erfindung zu gewähren. Entsprechend den kommerziellen Interessen einer Anmelderin sind die Patentschriften oft derart formuliert, dass sie einen möglichst hohen Schutzumfang versprechen und der Konkurrenz nur so viel praktische Hinweise zur Implementierung der betreffenden oder ähnlicher Technologien geben, wie es für die Erteilung des Patents erforderlich ist.

Maschinenlernende Entwurfsplattform

Ein äußerst allgemein gehaltenes Patent dieser Art ist 'Maschinenlernende Entwurfsplattform' der Taiwan Semiconductor Manufacturing Co. Ltd. [14] Es wurde am 10. Oktober 2017 beim Deutschen Patent- und Markenamt angemeldet und befindet sich derzeit Stand: (16. August 2019) noch in der Prüfung. Außerdem wird es in den USA, Taiwan und China ebenfalls noch geprüften. Es postuliert die Anwendung von maschinellem Lernen im Entwurfsprozess um die 'Time-to-Market', also die Zeit von Beginn der Entwicklung bis zum Verkaufsbeginn des Produkts, zu reduzieren. Der Entwurfsflow, die Computersysteme auf denen die Entwurfssoftware läuft und auch der Vorgang mit dem Daten aus dem Entwurf gewonnen werden sollen ('Merkmalsextraktion', *feature extraction*) sind in verschiedenen Ausführungen beschrieben. Sollte dieses Patent von den Patentämtern genehmigt werden, würden darunter wahrscheinlich alle bisher beschriebenen Verfahren zur Vorhersage und Korrektur von Entwurfsfehlern fallen.

Hotspot detection based on machine learning

Deutlich konkreter ist 'Hotspot detection based on machine learning' [52] von Mentor Graphics Corp. Seit dem Anmeldetag am 27. Mai 2014 wird es vom US Patent Office geprüft (Stand: 16. August 2019). Anmeldeverfahren laufen auch in China und Taiwan. Diverse trainierte maschinelles Lernen Modelle (*kernels*) sollen Schaltungstopologien analysieren um Stellen (*hotspot*) an denen der lithografische Prozess versagen könnte zu identifizieren. Für jede Art von Fehlerquelle wird ein dafür trainiertes Modell genutzt. Das in ihrem Implementierungsbeispiel verwendete Lernverfahren ist die Support Vector Machine(SVM).

Die vier von Yang et al. zusammengefügten 28nm Entwürfe sind im experimentellen Teil des Implementierungsbeispiels einzeln bearbeitet worden. Auffällt, dass im Gegensatz zur Hotspot Erkennung von Yang et al. keine Grafikkarten die hier verwendeten zwei Intel Xeon 2,3GHz Prozessoren mit 64GB Arbeitsspeicher unterstützen. Die aufaddierte CPU-Laufzeit der Patentimplementierung von 12 370s ist nur ein Fünftel der 60 147s, die das Verfahren von Yang et al. zur Analyse des kombinierten Designs benötigte. Diese geringeren Ressourcenanforderungen zeichnen SVM gegenüber Lösungen mit neuronalen Netzen aus. Punkten kann die auf neuronalen Netzen basierende Methode von Yang et al. mit 98,2% Genauigkeit, während die auf Support Vector Machinesbasierende Implementierung Werte zwischen 85,9% und 98,2% erzielt.

Identification of hot spots or defects by machine learning

Der Lithografie-Equipment Hersteller ASML verfolgt in seinem 2016 angemeldeten und am 16. Mai 2019 veröffentlichten Patent 'Identification of hot spots or defects by machine learning' [3] nicht nur nach Lithografie Hotspots, sondern auch nach Problemquellen für andere Prozesschritte. Als Eingangsdaten für ihr maschinelles Lernen Modell dienen hier unter anderen die Parameter der Prozesschritte, wie z.B. die Belichtungsdosis. Bestimmt wird bei welchen Prozesschritten, zum Beispiel Lithographie oder **CMP!** (**CMP!**), Defekte auftreten können. Die Autoren halten es sich offen welches maschinelle Lernen Modell, ob supervised oder unsupervised, verwendet werden kann Unsupervised Learning wird nur in einem anderen Patent zum Finden von Hotspots, IBMs 'Automated lithographic hot spot detection employing unsupervised topological image categorization' [69], erwähnt. Im Implemenentierungsbeispiel wird allerdings, wie in den anderen Patenten auch, SVM verwendet.

2.2.3 Weitere Anwendungen im Schaltungsentwurf

Maschinelles Lernen wird auch auf Fragestellungen des Schaltungsentwurfs angewendet, die nichts mit der Vorhersage von späteren Problemstellen zu tun haben.

'Machine-Learning based datapath extraction' [68] unterscheidet, ausgehend von einer Netzliste, ob eine Gruppe an Standardzellen Datenpfadlogik ist oder nicht. IBM meldete es 2012 an und übertrug es im Jahr der Erteilung 2015 an Globalfoundries Inc. Die Klassifizierung erfolgte sowohl mit Support Vector Machinesals auch mit neuronalen Netzen.

'System and method for designing a chip floorplan using machine learning' [31] nutzt vorhergesagte Werte für benötigte Fläche und Energieverbrauch als Feedback für Optimierungen beim Entwurf von Netzwerken auf dem Chip (*Network-on-Chip* / NoC). Seit der Anmeldung 2018 durch Arteris Inc. in den USA ist es in Prüfung.

3 Präzisierung der Aufgabenstellung

3.1 Ziel und Zweck

Anhand einer platzierten Netzliste soll eine Vorhersage getroffen werden an welchen Stellen es zu Problemen bei der Feinverdrahtung kommen wird.

3.2 Ausgangspunkt und Abgrenzung

Mit modernen Prozesstechnologien und stetig komplexer werdenden Schaltungen kann immer weniger davon ausgegangen werden, dass ein platziertes Layout, auch nach erfolgreicher Globalverdrahtung, im Detail verdrahtet werden kann. Da die Feinverdrahtung mit Abstand am meisten Rechenressourcen benötigt, sind Verfahren die Probleme schon nach der Platzierung feststellen können. Um solche Vorhersagen zu machen bieten sich Verfahren des maschinellen Lernens an. Diese Modelle sollen mit Informationen jeder Gitterzelle des platzierten Designs und Fehlermeldungen zu Kurzschlüssen aus der Feinverdrahtung in der betreffenden Gitterzelle trainiert werden.

3.3 Zu erreichende Arbeitsergebnisse

3.3.1 Stand der Technik

Durch Recherche in wissenschaftlicher Literatur und veröffentlichten Patenten soll der Stand der Technik bei der Anwendung von Methoden des maschinellen Lernen auf Probleme der Schaltungsverifikation ermittelt werden.

3.3.2 Passendes Datenmaterial

Gesucht sind kostenlos und unter einer für akademische Zwecke ausreichender Lizenz verfügbare Schaltungensentwürfe, die das notwendige Maß an Komplexität haben, dass zu untersuchende Verdrahtungsfehler auftreten. Am besten wäre Datenmaterial, welches in sich mit der gleichen Thematik wie diese Arbeit beschäftigenden wissenschaftlichen Arbeiten verwendet wurde. Dies ermöglicht vergleichbare Ergebnisse.

3.3.3 Entwurfswerkzeuge

Da Informationen sowohl aus der Platzierung als auch der Feinverdrahtung benötigt werden, sind geeignete Entwurfswerkzeuge für Platzierung und Verdrahtung notwendig. Nur im Fall,

dass passende Entwürfe in diesen Entwurfsphasen frei verfügbar sind, würden Platzierer und Verdrahter nicht benötigt. Auf jeden Fall notwendig sind Werkzeuge die aus den Entwurfsphasen Daten, visuelle oder anderweitige, extrahieren. Idealerweise sind all diese Werkzeuge unter einer Open-Source Lizenz verfügbar. Etwas weniger vorteilhaft sind Programme, die zwar kostenlos, aber ohne die Möglichkeit den Quellcode zu inspizieren, vertrieben werden. Kommerzielle Entwurfssoftware ist als letzte Option zu verwenden. Deren Nutzung ist selbstverständlich nur dann möglich, wenn das Institut für Feinwerktechnik und Elektronik-Design über entsprechende Lizenzen verfügt und die Software am besten schon installiert und konfiguriert ist. Die Arbeit soll einen Überblick über die infrage kommenden Programme, mit Vor- und Nachteilen für den Einsatz zur Synthese von Netzlisten frei verfügbarer VLSI Schaltungen, geben.

3.3.4 Datensätze

Die frei verfügbaren Schaltungsentwürfe müssen in die Datenbank der Entwurfswerkzeuge importiert werden und dann als platzierter und als detailliert verdrahteter Entwurf abgespeichert werden. Aus dem platzierten Entwurf müssen pro Gitterzelle Informationen extrahiert und gespeichert werden, die einen Hinweis auf die Verdrahtbarkeit geben. Zugeordnet zu den gleichen Gitterzellen soll die Anzahl der bei der Feinverdrahtung entstandenen Kurzschlüsse gespeichert werden. Diese Datensätze werden dann in ein Datenformat abgespeichert, das sich für Eingangsdaten der maschinelles Lernen Modelle eignet. Eine Übersicht über zu akademischen Zwecken frei verfügbare VLSI-Schaltungen soll gegeben werden. Besonderer Augenmerk liegt auf Datensätzen aus akademischen Wettbewerben. Die Datensätze müssen in einem Format vorliegen, das mit den ausgewählten Werkzeugen synthetisiert werden kann.
Die ausgewählten Datensätze und Entwurfswerkzeuge müssen miteinander kompatibel sein, hier besteht also eine gegenseitige Abhängigkeit der Teilergebnisse.

4 Entwurf

Ein großer Teil des Entwurfsprozesses war die Suche nach und Testen von Werkzeugen (Abschnitt 4.2) und Datenquellen (Abschnitt 4.1) zum Erzeugen von platzierten und verdrahteten Designs. Darauf folgte die Extraktion (Abschnitt 4.3.1) aussagekräftiger Eigenschaften der Gitterzellen eines Designs um mittels maschinellem Lernen Vorhersagen über die Verdrahtbarkeit des Layout zu generieren. Die Auswahl und Parametrierung eines geeigneten maschinelles Lernen Verfahrens ist in Abschnitt 4.3.2 beschrieben.

Der für die Ausführung verwendete Code steht ebenso wie eine PDF dieser Arbeit im Gitlab Repository [67] `https://gitlab.hrz.tu-chemnitz.de/IFTE-EDA/rpml` für Angehörige der Arbeitsgruppe zur Entwurfautomatisierung am Institut für Feinwerktechnik und Elektronik-Design der TU Dresden zur Verfügung.

4.1 Layoutdaten

Im Gegensatz etwa zu Software gibt es nur wenig kostenlos und offen verfügbare Layouts für Digitalschaltungen Hardware. Lösungen mit FPGA kommen für das Ziel dieser Arbeit, Kurzschlüsse anhand von platzierten Netzlisten vorhersagen, nicht infrage. Da FPGAs aus Arrays von Transistoren bestehen, deren Verbindung untereinander konfigurierbar ist (*field programmable*), kommt es normalerweise während der Synthese nicht zu Kurzschlüssen.

4.1.1 Open-Source Daten

OpenCores

Die größte Website für Open-Source IP (*Intellectual Property*) Cores ist *OpenCores.*[45] (IP Cores sind fertig entworfene Module, die zusammen mit anderen IP Cores und Eigenentwürfen in einem größeren Design verwendet werden.) Sie gehört der Digitaldesign Firma Oliscience BV.[8] Nach der kostenlosen Registrierung auf der Seite kann man dokumentierten Quellcode von verschiedenen Projekten herunterladen, zum Beispiel eine Schnittstelle für den I2C Kommunikationsbus Standard[26] oder der 16bit Mikrocontroller OpenMSP430[22]. Projektdateien werden mithilfe der Versionsverwaltungssoftware *Subversion* (SVN) auf den Server geladen und heruntergeladen. Es ist jeweils vermerkt, ob die IP bereits erfolgreich als anwendungsspezifischer Entwurf (*Application Specific Integrated Circuit*ASIC) auf Silizium (*silicon proven*) oder mithilfe von FPGAs (*field programmable Gate Array*) implementiert wurde.

Zum Beispiel bietet der bereits erwähnte OpenMSP430[22] Unterstützung für die Synthese als ASIC oder FPGA. Spezifische Synthese Dateien gibt es für die FPGA Werkzeugketten von Actel, Altera und Xilinx. Entsprechende Dateien zur Synthese eines auf Siliziumbasis fertigbaren

ASIC gibt es nur für den kommerziellen Closed-Source *Synopsis Design Compiler*. In diesem Stadium der Arbeit war das Ziel des Autors einen rein auf kostenlos verfügbarer und quelloffener Software und Hardwarebeschreibungen basierenden Entwurfsflow zu erstellen. Ein kommerzielles Synthese Werkzeug zu verwenden widerspricht dem. Mit der in Abschnitt 4.2.1 beschriebenen Open-Source Software konnte der Mikroprozessor trotz erfolgreicher Verdrahtung und Platzierung leider nicht erfolgreich synthetisiert werden. Das zentrale Problem war, dass die mit Qflow mitgelieferte Technologiedatei `osu18` nicht kompatibel mit den bisher erzeugten Standardzellen war. Dies verhinderte den Export der bisher erzeugten, Werkzeug eigenen, Dateiformate in das Standard Layout Format GDSII. Mit einigen Anpassungen im Quellcode sollte die Synthese dennoch möglich sein: Die Website von Qflow zeigt ein Bild des verdrahteten IP Cores mit dem Hinweis, dass er mittels Qflow platziert und verdrahtet worden sei.[51]

LibreCores

LibreCores[39] verfolgt einen ähnlichen Ansatz wie OpenCores, wird aber nicht von einem kommerziellen Unternehmen getragen, sondern von der gemeinnützigen *Free and Open Source Silicon Foundation* (FOSSi)[62].[2] Es stellt außerdem Projektdateien nicht auf der Website selbst zur Verfügung, sondern verweist auf Software Hosting Websites wie Github. Positiv ist, dass keine Registrierung zum vollen Zugriff auf die Entwürfe notwendig ist.

Auch auf LibreCores gibt es eine Reimplementierung des MSP430 Mikrocontrollers. Der Neo430[44] ist zwar in 'plattformunabhängigem VHDL' beschrieben, bietet explizite Unterstützung aber nur für FPGAs und nicht für die Implementierung als Silizium Chips.

Dieses Beispiel zeigt, dass es auch auf LibreCores interessante frei verfügbare Entwürfe gibt, diese aber nicht in dem Maße *silicon proven* sind wie die auf OpenCores.

4.1.2 Akademische Wettbewerbe

Um den Stand der Forschung im Bereich der Entwurfsautomatisierung voranzubringen, gibt es regelmäßige, für akademische Forscher offene, Wettbewerbe. Unterstützt werden sie meist von Unternehmen der Halbleiterbranche. Gerade Hersteller von Entwurfssoftware wie Synopsis oder von FPGAs samt zugehöriger Software Unterstützung wie Xilinx engagieren sich. Das Ziel dieser Wettbewerbe wird oft mithilfe von Software der beteiligten Unternehmen evaluiert. Zum Beispiel wird bei einem Wettbewerb um den besten Platzierer das platzierte Layout durch den kommerziellen Verdrahter des Unternehmens fertiggestellt und die Qualitäts-Metriken erfasst. Diese Aufgabe erfüllte bei den ISPD Wettbewerben 2014 und 2015 die Software *Olympus SOC* des damaligen Sponsors Mentor.

Das dabei zur Verfügung gestellte Datenmaterial (*benchmarks*) ist für die Forschung, auch ohne Teilnahme an den Wettbewerben, sehr wertvoll.

ISPD

Das *International Symposium on Phyical Design* ISPD befasst sich vor allem mit Problemen der Platzierung und Verdrahtbarkeit. Der dazugehörige Wettbewerb hatte bisher immer für zwei Jahre den gleichen Sponsor und damit auch das gleiche Thema.[27] 2010 und 2011 war das Ziel '*Routability-driven Placement*', also die Platzierung von Bauelementen unter besonderer Berücksichtigung der späteren Verdrahtbarkeit. Im Jahr 2012 und 2013 ging es, gesponsert von IBM um die Dimensionierung von Transistorgates.[28] 2014 und 2015 ging es, unterstützt durch Mentor, nochmals um Platzierung, diesmal mit besonderem Fokus auf die spätere Detailverdrahtung unter Berücksichtigung von Platzierungs- und Verdrahtungshindernissen (*blockages*) . [49] 2016 und 2017 unterstützte Xilinx Wettbewerbe zur Platzierung auf FPGAs. 2018 und 2019 war Cadence der Sponsor für initiale Detailverdrahtung.

ICCAD

Der ICCAD Wettbewerb befasst sich vor allem mit Lithographie Hotspots und stellt dafür Geometriedaten von fertigen Layouts mit Information zu den Orten der Hotspots bereit.[64] In Abschnitt 2.2.1 ist beschrieben wie Zhang et al. das Material nutzten um ihr neuronales Netz zu trainieren.[75]

TAU

Die Modellierung von zeitlichem Schaltkreisverhalten steht im Fokus des TAU Wettbewerbs.[60] Dazu werden Hardwarebeschreibungen in Verilog, Bibliotheken mit Timing der Bauelemente im Library File Format sowie zusätzliche Konfigurationsdateien von Beispielschaltungen bereitgestellt. Auch Software von in vergangenen Wettbewerben erfolgreichen Forschern steht zum Download bereit.

4.2 Entwurfswerkzeuge

4.2.1 Open-Source Werkzeuge

Verglichen mit der restlichen Open-Source Software Welt, etwa der Vielfalt von Programmiersprachen und Texteditoren, ist das Angebot an Werkzeugen für die Entwurfsautomatisierung recht übersichtlich.

Für FPGAs ist der Verdrahter und Platzierer *nextpnr*[24] das Open-Source Projekt mit der größten Community. Für die Synthese des Verilog-Codes wird das unten näher beschriebene *Yosys* eingesetzt. Als quelloffenes Backend um gerätespezifischen Bytecode zu generieren dient für die *Lattice iCE40* Modellreihe *Project IceStorm*[72]. Proprietäre Technologie-spezifische Daten und Werkzeuge stellen FPGA-Hersteller oft kostenlos zur Verfügung um ihre Umsätze anzukurbeln.

Die im Folgenden detaillierter beschriebene Software ist für den ASIC Entwurf geeignet. Sie ist zum Großteil in C und C++ verfasst und wird mit dem auf UNIX Systemen üblichen `make`, `make install` Flow installiert. *Qflow* koordiniert verschiedene Open-Source Entwurfswerkzeuge um ausgehend von einer Verilog Hardwarebeschreibung ein fertigbares Layout im GDSII Format zu erstellen.[51] Die dafür im Folgenden vorgestellten Werkzeuge sind, in der Beurteilung des Autors, die für ihren jeweiligen Aufgabenbereich beste verfügbare Open-Source Software. Der Großteil davon ist, ebenso wie *Qflow* selbst, auf der Website von Tim Edwards 'Open Circuit Design'[19] und auf Github[21] erhältlich. Open Circuit Design wird von Tim Edwards' Arbeitgeber, der Halbleiter Designfirma Efabless, unterstützt.

Proton

Das direkt von Efabless stammende *Proton*[50] implementiert, genauso wie *Qflow*, einen *ASIC Place and Route Flow* mit Import und Export von gebräuchlichen Chip-Datenformaten wie Verilog, LEF, DEF und GDSII. Laut den *Commit* Nachrichten auf Github arbeitet es inzwischen auch auf Basis der qflow Software-Engine, wurde jedoch in Perl und nicht C geschrieben. Auf einer aktuellen Linux Distribution (Fedora 30) war es schwierig alle für *Proton* notwendigen Perl-Bibliotheken zu installieren. Deswegen wurde im Folgenden nur *Qflow* verwendet.

Qflow

Qflow beherrscht (Stand 2018) Entwürfe mit bis zu 30 000 Gates und kann mit Entwurfsregeln für Technologieknoten von ca. 65nm und älter umgehen.[63, S.11] Kompliziertere Entwurfregeln, wie von Alpert et al. in [5] für 32nm beschrieben (siehe Abschnitt 2.1.1), können mit Open-Source Programmen also nicht untersucht werden. Dies limitiert deren Nützlichkeit, um durch Detailverdrahtung entstandene Kurzschlüssen zu finden und vorherzusagen, stark.

Verilog Synthese

Yosys ist ein Werkzeug um in Verilog-2005 verfasste Hardware-Beschreibungen in RTL (*Register Transfer Logic*) zu synthetisieren.[71] Das Ausgabeformat ist BLIF (*Berkeley Logic Interchange Format*).

Platzierer

Graywolf ist ein Fork, also eine Abspaltung vom ursprünglichen Entwicklungszweig, der *Timber-Wolf* Software. *Timberwolf* wurde an der Yale Universität, ursprünglich quelloffen, entwickelt. *Graywolf* wurde von der letzten unter einer Open-Source Lizenz stehenden Version 6.3.5 abgespalten. Diese konnte platzieren und global verdrahten, aber nicht detailverdrahten. Seitdem wurde primär das Verhalten als Linuxprogramm verbessert und nicht die Kernlogik. [54]

Verdrahter

Da es keinen Open-Source Detailverdrahter für seinen *Qflow* Designflow gab, programmierte Tim Edwards dafür den *Qrouter*[20] selbst. Nach eigenen Angaben ist der Verdrahter nur 'moderately capable', kommt also nicht an die Leistung branchenüblicher kommerzieller Verdrahter heran.

Das Eingabeformat ist die Kombination aus LEF (*library exchange format*) und DEF (*design exchange format*) Dateien. Abgespeichert wird das verdrahtete Layout als DEF Datei.[51] Es ist also möglich dieses Programm mit Werkzeugen jenseits des *Qflows* zu kombinieren.

Klayout

Das beliebteste Open-Source Werkzeug um Layoutdateien anzuzeigen, zu editieren und Entwurfsregeln zu kontrollieren ist *Klayout*[35]. Er kann durch Skripte in den Programmiersprachen Python oder Ruby gesteuert werden. Es verarbeitet Layoutdateien in den Formaten GDSII und OASIS.[34]

Anfangs war der Ansatz dieser Studienarbeit ein maschinelles Lernen Modell anhand visueller Layoutdaten zu trainieren. Für die automatisierte Erzeugung von nach Layout-, bzw. Metallschichten (*metal layers*) getrennten und in Gitterzellen unterteilten Bilddateien war *Klayout* von großem Nutzen. Das Skript dafür steht im oben erwähnten Gitlab Repository als `Klayout/klayout_scripts.py` zur Verfügung.

4.2.2 Akademische Closed-Source Werkzeuge

Für die verschiedenen Wettbewerbe im Bereich Digital Design entstanden viele Programme, vor allem Platzierer und Globalverdrahter. Einzig die Detailverdrahtung ist offenbar ein so komplexes Problem, dass es bisher keine Wettbewerbe für vollständige Detailverdrahtung und damit entsprechende Werkzeuge von akademischen Forschern gab. Die Teilnahmebedingungen der Wettbewerbe schreiben meist die Einreichung von auf Linux-Servern mit x86-Prozessoren lauffähigen Binärdateien vor. Daher werden die Programme ohne Einblick in den Quellcode verteilt. Aufschlüsse über die Funktionsweise geben die begleitenden wissenschaftlichen Artikel.

Eine Schwierigkeit ist, dass es trotz Hinweisen auf die Software durch Bestenlisten bei Wettbewerben und wissenschaftlichen Arbeiten oft keine Möglichkeit gibt die Software zu erhalten. Einige wenige Programme werden dankenswerterweise der Öffentlichkeit zur Verfügung gestellt.

Nach Registrierung auf einem Webformular ist das Analysewerkzeug für globale Verdrahtungsengpässe *CGRIP*[56] frei herunter ladbar[23]. Der für diese Arbeit benutzte *Eh?Placer*[43] wurde dem Autor, nach einer Email mit den auf seiner Website[17] verlangten Angaben, zugesandt. Den Globalverdrahter mit Konvertierfunktion vom Wettbewerbsformat des ISPD08 in LEF/DEF *NCTU-GR* kann, für den akademischen Gebrauch lizenziert, einfach herunterladen[41].

DATC Robust Design Flow

Auch im Bereich der akademischen Platzierer gibt es Bemühungen einen durchgehenden Designflow zu schaffen. Der *DATC Robust Design Flow*[29] lädt Benchmarks vom TAU Contest 2017 [60] zum Thema Makro Timing Analyse herunter und ergänzt sie mit Dateien aus anderen Wettbewerben um insgesamt 26 vollständige Benchmark Entwürfe zu erhalten.[30] Für jeden der Entwurfsschritte, von Logiksynthese bis zur Detailverdrahtung, gibt es Skripte die spezifische Werkzeuge steuern. In einer Datei können Parameter, wie verwendetes Werkzeug oder Zieldichte, für den Entwurfsflow geändert werden.

Ein Mangel ist, dass bisher die Werkzeuge für die einzelnen Entwurfsschritte nicht zusammen mit dem DATC Robust Design Flow ausgeliefert werden. Die im Folgenden beschriebene Software wird auch in diesem Designflow verwendet und muss bei den jeweiligen Urhebern besorgt werden.

Platzierer

Für Platzierer sind die ISPD Wettbewerbe von 2014[48] und 2015[49] besonders relevant. Sehr gut abgeschnitten hat in beiden Jahren der *Eh?Placer* mit dem ersten Platz 2014 und dem zweiten Platz 2015. Da es der einzige leicht verfügbare Platzierer war und zudem bereits erfolgreich von den Urhebern der Software in anderen wissenschaftlichen Arbeiten [58] verwendet wurde, fiel die Entscheidung für diese Arbeit auf den Eh?Placer.

Verdrahter

Es existieren einige akademische Globalverdrahter, wie zum Beispiel der NCTU-GR 2.0[41]. Allerdings sind diese ohne Detailverdrahtung nur zum Ermitteln von Parametern der Globalverdrahtung (siehe Abschnitt 2.1.1) zu gebrauchen und nicht zur vollständigen Verdrahtung. Wie bereits erwähnt gibt es so gut wie keine performanten Detailverdrahter im akademischen, kostenfreien Bereich. Daher musste hier mit kommerziellen Werkzeugen gearbeitet werden.

4.2.3 Kommerzielle Werkzeuge

Es gibt drei dominierende Hersteller von Software zur Entwurfsautomatisierung: Cadence, Mentor und Synopsis. Diese bieten jeweils Software für alle Schritte des Entwurfsablaufs an. In den für diese Arbeit besonders relevanten ISPD 2014 und 2015 wurde die *Olympus SOC* Software von Mentor zum Verdrahten der von den Teilnehmern platzierten Layouts und der Beurteilung der Qualität der Platzierung eingesetzt. Es hätte sich also für maximale Kompatibilität angeboten auch diese Software zur Verdrahtung und Extraktion von Daten zum platzierten Layouts und dem Vorkommen von Kurzschlüssen zu verwenden. Jedoch ist am Institut für Feinwerk Technik und Elektronik Design die *Encounter Digital Implementation* Suite lizenziert und installiert, daher wurde sie in dieser Arbeit benutzt.

Platzierer

Encounter hätte auch diesen Entwurfsschritt erledigen können, dennoch wurde der oben erwähnte Eh?Placer vorgezogen, da er frei verfügbar war und auf die Bedingungen der ISPD 2014 und 2015 Wettbewerbe angepasst.

Verdrahter

Der in Encounter enthaltene *nanoRoute* Verdrahter war für die Komplexität der Benchmarks am Besten geeignet. Es erledigte Global-und Detailverdrahtung und wurde nur mit minimaler Konfiguration (siehe Quelltext 4.1) ausgeführt.

Quelltext 4.1: Auszug aus `Encounter/route.tcl`

```
1 # Routing should not be timing driven since ISPD2015 has no timing constraints
2 setNanoRouteMode -drouteAutoStop false
3 globalRoute
4 detailRoute
```

4.3 Neuronales Netz

Wie aus Kapitel 2 zum Stand der Technik hervorgeht, wurde in der Literatur entweder die Stützvektormethode (SVM) oder ein neuronales Netz als Verfahren zur Vorhersage von Verdrahtbarkeit mittels maschinellem Lernen eingesetzt. Für diese Arbeit wurde ein neuronales Netz verwendet, da es im Gegensatz zu SVM flexibler handhabbar ist, und zum Beispiel auch Bilddaten als zusätzliche Eingangsdaten gut verarbeiten kann. Es kann zudem stetig mit neuen Daten verbesser werden (*online learning*).

4.3.1 Feature Extraktion

Da es, wie oben bei im Abschnitt zu Klayout beschrieben, doch nicht praktikabel war das neuronale Netz mit Bilddaten zu füttern, mussten aus den Layoutdaten für jeden Sektor Eigenschaften (Features) extrahiert werden, die einen Hinweis auf die spätere Verdrahtbarkeit geben können. Die Abfrage der Informationen geschah in der zur Verdrahtung benutzten Encounter Software.

Übersicht über bisher verwendete Features

In der Literatur werden als relevante Eigenschaften (*features*) von Tabrizi et al. die relative Position der Gitterzellen, die Belegungsdichte mit Entwurfszellen des gesamten Entwurfs, die Zahl der lokalen und globalen Netze, *narrow channel* Phänomene, die Verteilung der Anschlusskontakte in einer Gitterzelle, die Anzahl der Anschlüsse für das Clock Netzwerks, die Belegung von vertikal, bzw. horizontal verlaufenden Verdrahtungskanälen und die Anzahl der Pins von nicht standardmäßigen Netzen genannt.[58] Chan et al. berücksichtigten ausserdem Entwurfszellen die erfahrungsgemäß zu Problemen in der Gitterzelle führen, strukturelle Parameter wie die Anzahl und Art von an Ein- und Ausgang (*fanin* und *fanout*) verbundenen Zellen sowie Erkenntnisse aus der Globalverdrahtung wie den Overflow an den Kanten der Gitterzellen.[10] Zudem verwendeten Zhou et al. nutzten zudem das Aufkommen von Verdrahtungsblockaden und wiesen darauf hin, dass bei der Verwendung zu vieler Parameter Over-Fitting auftreten kann, das Netzwerk also nur die Zuordnung zu den gerade gesehenen Gitterzellen lernt und nicht generalisiert. [76]

Im Folgenden wurde nur eine kleine Auswahl dieser Features verwendet, insgesamt acht Werte pro Gitterzelle. Der Vorteil all diese Werte pro Gitterzelle zu erfassen und auf deren Grundlage Probleme für jede einzelne Gitterzelle vorherzusagen ist einerseits, dass Kurzschlüsse besser lokalisiert und behoben werden können, und andererseits, dass so trotz der Knappheit an geeigneten Layouts genug Datenproben (Samples) für effektives maschinelles Lernen entstehen. Sie werden so ausgewählt, dass sie sowohl aussagekräftig als auch vergleichsweise einfach zu extrahieren waren. Jedes Trainingssample beinhaltete neben der betrachteten Zelle selbst auch ihre acht Nachbarzellen. Die resultierende Struktur einer 8x9 Matrix ist in 4.1 dargestellt.

Größe der Gitterzellen

Genauso wie bei [58] wurde die Größe der einzelnen Gitterzellen auf eine Seitenlänge von drei Reihen (*rows*) festgelegt. Dies entsprach bei den untersuchten Layouts 6 µm. Der Zellindex in x-Richtung war `i` und der in y-Richtung `j`.

	Koordinate	numInsts	InstPins	LocalNets	GlobalNets	Pins_stddev_x	Pins_stddev_y	instsArea	dist_center
Zentrum	(i, j)	x	x	x	x	x	x	x	x
Oben	(i, j+1)	x	x	x	x	x	x	x	x
Oben-Rechts	(i+1, j+1)	x	x	x	x	x	x	x	x
Rechts	(i+1, j)	x	x	x	x	x	x	x	x
Unten-Rechts	(i-1, j-1)	x	x	x	x	x	x	x	x
Unten	(i, j-1)	x	x	x	x	x	x	x	x
Unten-Links	(i-1, j-1)	x	x	x	x	x	x	x	x
Links	(i-1, j)	x	x	x	x	x	x	x	x
Oben-Links	(i-1, j+1)	x	x	x	x	x	x	x	x

Abbildung 4.1: Struktur eines Trainingssample mit den Werten für die Zahl der Instanzen, der Anschlusskontakte, lokalen und globalen Netzen, Standardabweichung der Anschlusspositionen in x- und y-Richtung, der Gesamtfläche der Instanzen einer Gitterzelle und des Abstandes der Gitterzelle vom Zentrum des Entwurfs.

Verteilung der Pins

Relevant für die Verdrahtbarkeit ist die Verteilung der Pins. Dicht beieinander liegende Pins können schwer bis unmöglich zu verdrahten sein, auch wenn die durchschnittliche Anschlusskontaktdichte gering ist. Daher wurde die Standardabweichung der x- und y-Positionen der Anschlusskontakte in jeder Zelle berechnet.

Position der Gitterzelle

Da Gitterzellen im Zentrum eines Designs wegen der Vielzahl von sie kreuzenden Verbindungen schwieriger zu verdrahten sein können als solche am Rand, wurde die relative Entfernung d der Gitterzellen von der Mitte des Layouts $i_m = \frac{i_{max}}{2}$, bzw. $j_m = \frac{j_{max}}{2}$ berechnet.

$$d = \sqrt{(i - i_m)^2 + (j - j_m)^2} \tag{4.1}$$

Globale und lokale Netze

Für die Verdrahtbarkeit ist es relevant, ob die Anschlusskontakte in einer Gitterzelle nur untereinander verbunden werden müssen, also zu einem lokalen Netz gehören, oder Verbindungen mit Pins in anderen Gitterzellen haben. Die Definition eines lokalen Netzes mit mindestens zwei lokalen Anschlusskontakten und eines globalen Netzes mit je mindestens einem lokalen und globalen Anschlusskontakt wurde von Tabrizi et al. übernommen. [58]

Verletzungen der Entwurfsregeln im verdrahteten Layout

Um Verletzungen der Entwurfsregeln zu erfassen wurde die Encounter Datenbank für jede Gitterzelle abgefragt nach allen Objekten des Typs `marker`. Ein Python Skript filterte dann aus diesen verschiedenen Meldungen die Kurzschlüsse (`Short`) heraus. Um das Trainieren des neuronalen Netzes zu vereinfachen, wurde jede Zelle mit Kurzschlüssen unabhängig von deren Anzahl als 'Treffer' (positiv) gewertet und Zellen ohne solche Fehler als 'Nieten' (negativ) gewertet. Die Existenz von Kurzschlüssen (`0` oder `1`) bildete das Label, also den Wert, den das Netz für jede Gitterzelle vorhersagen sollte.

Quelltext 4.2: Auszug aus `Encounter/get_info.tcl` zur Erfassungen von Entwurfsregelverletzungen

```
1 set markers_cur [dbQuery -area $box_cur -objType {marker}]
2 # write the subtypes of all markers of a tile into a csv
3 set marker_subtypes ""
4 for {set i 0} {$i < [llength $markers_cur]} {incr i} {
5 lappend marker_subtypes [dbGet -i $i $markers_cur.subtype]
6 }
```

Vorbearbeitung der Trainings- Validierungs- und Testdaten

Es gilt als vorteilhaft die erzeugten Daten auf ihr arithmetisches Mittel und Standardabweichung zu normieren, damit das Netzwerk nicht mit Eingängen aus weit auseinander liegenden Wertebereichen umgehen muss. [12] Im Laufe der Experimente mit unterschiedlich vielen Trainingsepochen und sowohl normiertem als auch unverändetertem Datenmaterial stellte sich aber heraus, dass die besseren Ergebnisse ohne Normierung erzielt werden. Daher wurde letztendlich keine Normierung der Eingangsdaten vorgenommen. Die Daten aus allen Layouts, abgesehen von `mgc_fft_2`, werden zusammengefasst. Davon dienten 80% zum direkten Trainieren des Netzes (`x_val`) und die restlichen Daten zum Validieren des Trainingserfolgs (`x_val`). Das Layout `mgc_fft_2` war weder Teil des Trainings noch der Validierung und wurde zum Testen des fertigen neuronalen Netzes genutzt.

4.3.2 Software Implementierung

Als *Framework*, also Werkzeugkasten, zur Konfiguration des Modells wurde das auf der Skriptsprache Python basierende Keras [13] genutzt. Dieses kann verschiedene *backends*, also auf mathematische Operationen für die Implementierung von neuronalen Netzen spezialisierte Werkzeuge, steuern. Für diese Arbeit wurde Tensorflow [1] von Google eingesetzt.

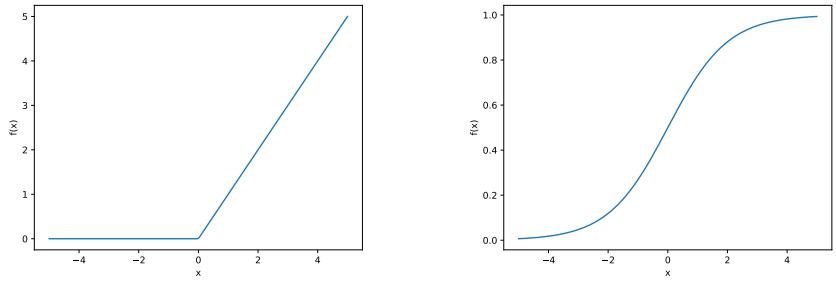
Netzwerkparameter

Neben der Gewinnung der Eingangsdaten ist die Wahl einer geeigneten Konfiguration für das Netzwerk, auch Hyperparameter genannt, essentiell für die Vorhersagekraft. Quelltext 4.3 zeigt wie das Modell im Keras Framework implementiert wurde.

Aufgrund des Ungleichgewichts in den Eingangsdaten, mit viel mehr fehlerfreie Gitterzellen als Zellen mit Kurzschlüssen, war es vorteilhaft, das Modell mit nur wenigen großen Chargen (*batch*) zu trainieren. Zu kleine Chargen können keine oder nur sehr wenige Gitterzellen mit Kurzschlüssen enthalten und sind daher nicht repräsentativ für die Gesamtheit. Gewählt wurde eine Größe von 10 000 Samples pro Batch.

Die Struktur ist der des Netzwerks in der Einleitung, Abbildung 1.2a, sehr ähnlich. Die Schichten sind sequentiell angeordnet, daher eine Instanz der Klasse `keras.models.Sequential`. Die Eingangsschicht und die verborgene Schicht haben jeweils 16 Knoten, die Matrizen der Dimension `input_shape` verarbeiten.

Die Aktivierungsfunktionen `relu` (*rectified linear unit*) und `sigmoid` in Abbildung 4.2 werden auf die Ausgänge der jeweiligen Schichten angewandt. Der Gleichrichter `relu`(siehe Abbildung 4.2a) bewirkt durch das auf null Setzen negativer Werte, dass auch nichtlineare Funktionen durch die Gewichte in den Eingangs- und verborgenen Schichten dargestellt werden können. In der Ausgangsschicht bildet die Sigmoidfunktion, zu sehen in Abbildung 4.2b den reellen Zahlenraum auf einen Wert zwischen 0 und 1 ab um so eine Wahrscheinlichkeit für das Auftreten einer Klasse, also das Vorhandensein von Kurzschlüssen, darzustellen.



(a) relu $f(x) = |x|$ für $x \geq 0$, sonst $f(x) = 0$

(b) Sigmoidfunktion $f(x) = \frac{1}{1+e^{-x}}$

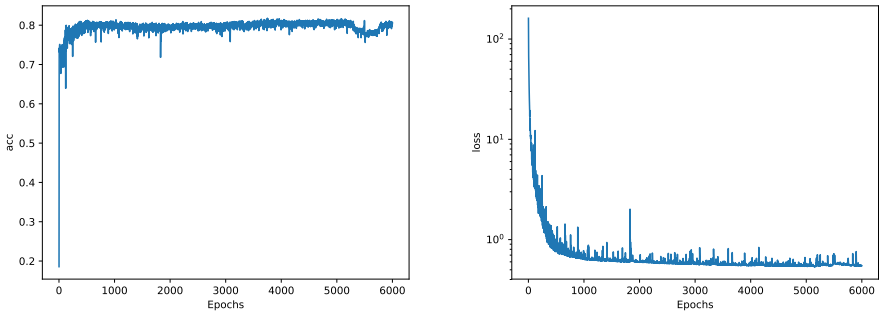
Abbildung 4.2: Aktivierungsfunktionen

Anhand der Eingangsdaten `x_train` und `x_val` und der 'wahren' Ausgangsdaten `y_train` und `y_val` passt das Modell in der Trainingsphase (`model.fit(...)`, Z.20-25) die internen Gewichte so an, dass eine Verlustfunktion minimiert wird. Der verwendete Optimierungsalgorithmus `Nadam`[18] ist eine Erweiterung des viel benutzten Adam [33] Algorithmus.

Die Verlustfunktion `binary_crossentropy` bestimmt die Entropie, also den Unterschied, zwischen der wahren und der vorhergesagten Werteverteilung.[6] Die 25fache Gewichtung der Ergebnisklasse `1`, also Gitterzellen mit Kurzschlüssen, wurde gewählt, da in den zum Training verwendeten verdrahteten Entwürfen etwa 25 mal so viele fehlerfreie Sektoren wie problematische Sektoren auftraten. Diese Gewichtung ist essentiell um brauchbare Ergebnisse zu erzielen, da das Netzwerk ansonsten durch die ausschließliche Vorhersage von fehlerfreien Sektoren in 96% der Fälle richtig läge. Eine Genauigkeit von weit über 90% weist ohne weiteren Kontext also nicht zwingend auf zufriedenstellende Vorhersagen hin.

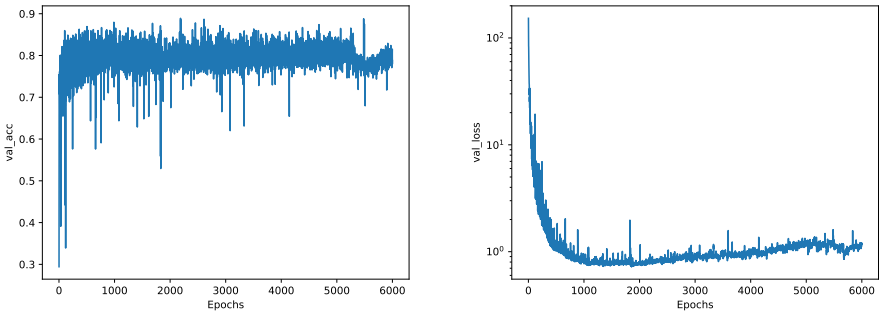
Wichtig ist auch über wie viele Epochen das Netz trainiert wird, also wie oft also mittels Backpropagation des aktuellen Ergebnisses der Fehlerfunktion die Gewichte in den einzelnen Schichten angepasst werden. Am Anfang steigt die Genauigkeit (*accuracy*), bzw. sinkt der Verlust (*loss*), mit jeder Iteration sowohl bei Vorhersage der Trainingsdaten, als auch bei der Validierung anhand von während des Trainings nicht gesehenen Daten. Die stetig wachsende Genauigkeit bei Vorhersage der Trainingsdaten zeigt Abbildung 4.3. Ab einem gewissen Punkt tritt jedoch Over-Fitting auf. Das Netzwerk lernt jedem Element der Trainingsdaten die richtigen Ausgangsdaten zu zuordnen, verallgemeinert dadurch aber nicht mehr so gut. Dadurch sinkt die Genauigkeit bei den Validierungsdaten. Abbildung 4.4a zeigt solch einen Verlauf der Validierungsgenauigkeit. Zu Beginn steigt die Genauigkeit sehr schnell, stagniert dann und sinkt am Ende leicht. Gut zu sehen ist auch die stochastische Natur des Lernverfahrens, besonders bei der Vorhersage von unbekannten Daten. Immer wieder gibt es Ausreißer nach oben oder unten ohne ein klares Muster. Der halb-logarithmische Graph des Verlusts (*loss*) in Abbildung 4.4b zeigt bei Validierungsdaten deutlich das schnelle Absinken bis auf ein Minimum bei ungefähr 1500 Epochen und danach den langsamen Anstieg durch Over-Fitting.

Daher werden für die Beurteilung der Ergebnisse in Kapitel 5 geeignete Metriken verwendet um Aufschluss über die Vorhersagekraft zu erlangen.



(a) Genauigkeit (b) Verlust (*loss*).

Abbildung 4.3: Training



(a) Genauigkeit (b) Verlust (*loss*)

Abbildung 4.4: Validierung

Quelltext 4.3: Keras Neuronales Netzwerk

```
def short_model(x_train, y_train, x_val, y_val):
    """Run a Keras sequential Neural Network.
    Args:
    x_train (numpy array):                Training features
    y_train (numpy array):                Training labels
    x_val (numpy array):                  Validation features
    y_val (numpy array):                  Validation labels
    Returns:
    history [keras.callbacks.History]: Data about training history
    model   [keras.models.Sequential]: Fitted Neural Network
    """
    model = Sequential()
    model.add(Dense(16, input_shape=(x_train.shape[1],), activation='relu'))
    model.add(Dense(16, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(optimizer='Nadam',
    loss='binary_crossentropy',
    metrics=['acc', 'binary_crossentropy'])

    history = model.fit(x_train, y_train,
    batch_size= 20000,    #int(y_train.shape[0] * 0.1),
    epochs=6000,
    validation_data=[x_val, y_val],
    verbose=1,
    class_weight={0:1.0, 1:25.0}
    )
    return history, model
```

4.3.3 Hardware Beschleuniger

Aufgrund der Vielzahl an benötigten Matrixmultiplikationen um die Gewichte eines neuronalen Netzes zu ändern genügen die Rechenressourcen einer herkömmlichen CPU (*Central Processing Unit*) mit vor allem linearer Abarbeitung der mathematischen Operationen nicht für komplexe neuronale Netzwerke.

Für die massiv-parallele Berechnung von Matrix-Operationen eignen sich Grafikkarten und anwendungsspezifische integrierte Schaltkreise (*Application Specific Integrated Circuits*) für neuronale Netze mit einer Vielzahl an Recheneinheiten deutlich besser.

Es wurde versucht das neuronale Netz auf Google Colab (`https://colab.research.google.com/`) [70] mit TPU (*Tensor Processing Unit)* Unterstützung laufen zu lassen. [15] Die TPU wurde von Google entwickelt um die Matrix- bzw. Tensoroperationen ihrer Tensorflow Engine möglichst effizient auszuführen. Allerdings unterstützt das verwendete Tensorflow 1.14 im Modul `tf.distribute.strategy` nicht die Gewichtung von Parametern (`class_weight`) In Tensorflow 2.1 wird es diese Einschränkung wahrscheinlich nicht mehr geben, allerdings war die TPU Unterstützung des im August 2019 aktuellen Release Candidate Tensorflow 2.0-RC noch nicht ausreichend um das zu testen.[61] Es sind zusätzliche Anpassungen von Nöten um eine TPU optimal zu nutzen, so muss zum Beispiel darauf geachtet werden, dass jede Charge (`batch`) ein Vielfaches von 1024 groß ist um die 8 Kerne mit jeweils 128 Recheneinheiten (*Execution Units* EU) auszulasten.

Solche Modifikationen sind bei Beschleunigung mithilfe einer Grafikkarte ((*Graphics Processing Unit* GPU) nicht nötig. Sie war mit Tensorflow 2.0 unkompliziert nutzbar. Zu beachten ist, dass ein Browserfenster mit dem Jupyter Notebook während der Berechnungen geöffnet bleiben muss und auf dem Client Rechner CPU Last erzeugt und, wenn viele Trainingsepochen durchgeführt werden, auch viel Arbeitsspeicher belegt. Nur auf CPU (Intel i5-3320M CPU mit zwei Kernen und 2,6 GHz Taktfrequenz) konnten zwei Trainingsepochen mit jeweils 120 000 Samples pro Minute, also etwa 4000 Samples pro Sekunde durchgeführt werden. Im auf Colab gehosteten Notebook mit GPU Unterstützung verarbeitete das neuronale Netz pro Minute ungefähr 400 Trainingsepochen, konnte also ca. 800 000 Samples pro Sekunde verarbeiten. Diese 200fache Beschleunigung hat es dem Autor erst erlaubt verschiedene Hyperparameter, auch solche mit vielen Wiederholungen (*epochs*), auszuprobieren. Das verwendete Jupyter Notebook `Keras/neural_network_GoogleColab_GPU.ipynb` steht im oben erwähnten Gitlab Repository [67] bereit.

5 Auswertung

Aufgrund von Inkompatibilitäten beim Verdrahten der Benchmark Layouts mittels der *Encounter* Software konnten leider nicht alle Entwürfe des ISPD 2015 Wettbewerbs untersucht werden. Wahrscheinlich liegt das daran, dass die Dateien auf den im damaligen Wettbewerb verwendeten *Mentor Olympus SOC* Verdrahter zugeschnitten waren.

Tabelle 5.1 zeigt die Vorhersage Ergebnisse, die ein wie in Kapitel 4.3 beschrieben konfiguriertes und trainiertes neuronales Netz, bei den jeweiligen Layouts erreicht hat. Da wie bereits erwähnt bei so ungleich verteilten Daten mit deutlich mehr 'Nieten' als 'Treffer' ein einziger Wert für die Genauigkeit wenig aussagekräftig ist, werden basierend auf einer Wahrheitsmatrix (*confusion matrix*) (Abbildung 5.1) die folgenden Metriken berechnet. Tabrizi et al. verwendeten diese Metriken auch in ihrer Arbeit zur Vorhersage von Verdrahtungsfehlern. [58]

Jede Layout hat n Gitterzellen mit TP Zellen mit Kurzschlüssen. Die True Positive Rate $TPR = \frac{TP}{TP+FN}$, auch Empfindlichkeit (*sensitivity*) genann, gibt an welcher Anteil der tatsächlich vorhandenen Kurzschlüsse gefunden werden. Abbildung 5.3a zeigt die Korrelation zwischen einer hohen Anzahl an vorhandenen Kurzschlüssen und einer hohen Trefferquote. Eine mögliche Erklärung dafür ist, dass bei Layouts mit nur wenigen problematischen Stellen eine kleine Anzahl an verpassten Stellen die prozentuale Genauigkeit stark senkt.

Analog dazu ist die Specificity $SPC = \frac{TN}{TN+FP}$, also der Anteil der korrekt gefundenen problemfreien Stellen gerade bei wenig problembehafteten Entwürfen besonders hoch. Die False Positive Rate $FPR = \frac{FP}{FP+TN}$, also der Anteil der fälschlicherweise für problemfrei erklärten Sektoren gemessen an allen tatsächlich problemfreien Stellen, ist dementsprechend bei kleineren Entwürfen besonders groß. Die Genauigkeit (Accuracy) $ACC = \frac{TP+TN}{n}$, (siehe Abbildung 5.2b) der Anteil der wahren Vorhersagen an der Gesamtheit, erreicht nur bei einigen wenigen Entwürfen mit wenigen Kurzschlüssen Werte über 90%. Wie bereits erwähnt kann aber gerade dieser Wert irreführend sein, da bei viel mehr Nullen als Einsen die ausschließliche Vorhersage von Nullen große Genauigkeit bringt. Dieser Sachverhalt bewirkt bei dem Entwürfen *mgc_pci_bridge32_b* mit sehr viel mehr fehlerfreien Gitterzellen als Zellen mit Kurzschlüssen eine große Genauigkeit von 96% während die Empfindlichkeit der Vorhersage mit 41% gering ist.

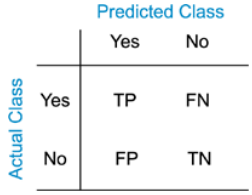


Abbildung 5.1: Wahrheitsmatrix [46]

Benchmark	n	TP	TPR	SPC	FPR	ACC	MCC
mgc_des_perf_1	5625	1455	0.966	0.333	0.667	0.497	0.300
mgc_des_perf_a	22500	1552	0.840	0.809	0.191	0.811	0.387
mgc_des_perf_b	10000	7	0.571	0.512	0.488	0.512	0.004
mgc_fft_1	2025	154	0.864	0.444	0.556	0.476	0.165
mgc_fft_2	3249	119	0.420	0.608	0.392	0.601	0.011
mgc_fft_a	17956	44	0.409	0.955	0.045	0.953	0.086
mgc_matrix_mult_1	8464	2082	0.943	0.349	0.651	0.495	0.281
mgc_matrix_mult_a	62500	695	0.652	0.917	0.083	0.914	0.209
mgc_pci_bridge32_a	4489	136	0.574	0.709	0.291	0.705	0.106
mgc_pci_bridge32_b	17956	17	0.412	0.960	0.040	0.960	0.058

Tabelle 5.1: Genauigkeitsmetriken für alle getesteten Layouts nach 3000 Trainingsepochen, wobei `mgc_fft_2` vom Training ausgeschlossen war.

Der Matthews Korrelationskoeffizienten $MCC = \frac{(TP*TN-FP*FN)/}{\sqrt{(TP+FP)*(TP+FN)*(TN+FP)*(TN+FN)}}$ hat einem Wertebereich von $[-1, +1]$, wobei $+1$ der beste Wert ist, und gilt gerade bei ungleichen Binärverteilungen als bester Indikator für die Vorhersagekraft.[11] Abbildung 5.3b zeigt, dass die besten Werte bei Entwürfen mit vielen problematischen Sektoren erzielt werden.

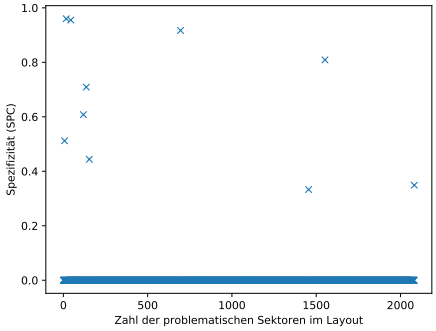
Das vom Training ausgeschlossene und als Testdaten verwendete `mgc_fft_2` Layout zeigt unterdurchschnittliche Werte für die Empfindlichkeit mit 42% und den Matthews Korrelationskoeffizienten mit 0.011. Ein Teil dieser schlechten Vorhersageleistung kann dadurch erklärt werden, dass es vergleichsweise wenige problematische Sektoren hat. Aber auch, dass das neuronale Netz dieses Layout weder als Trainings- noch als Validierungsdaten zur Verfügung hatte dürfte eine Rolle spielen.

Auch wenn die Vorhersagekraft des Modells nicht bei allen Layouts zufriedenstellend ist, zeigt es seine Stärken gerade bei Layouts mit vielen problematischen Gitterzellen. Vereinzelt auftretende Kurzschlüsse lassen sich leicht beheben. Bei schwerer zu verdrahtenden Layouts sind Vorhersagen zur Verdrahtbarkeit aber sehr wertvoll um die Platzierung iterativ, ohne rechenintensives Detailverdrahten, optimieren zu können. Während es zum Beispiel knapp drei CPU-Stunden dauerte das `mgc_matrix_mult_a` Layout zu verdrahten, benötigte die Vorhersage von Verdrahtungsfehlern nur wenige Sekunden.

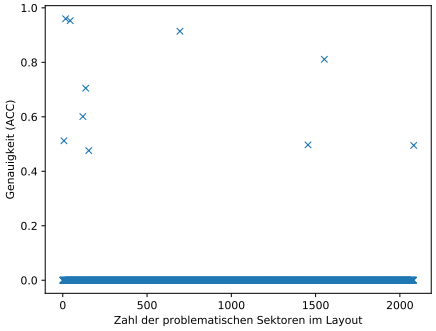
Die Korrelation zwischen Genauigkeit, bzw. Empfindlichkeit des Modells und Anzahl der Kurzschlüsse trat auch bei Tabrizi et al. auf. Sie erzielten aber im Durchschnitt deutlich bessere Ergebnisse mit, abgesehen von einem Ausreisser, einer Richtig-Positiv Rate von 68% bis 97%. Bei besonders guten Empfindlichkeitswerten sank aber, ebenso wie in dieser Studienarbeit, gleichzeitig die Spezifizität.[58]

Zhou et al. gaben bei ihren Vorhersagen der Verdrahtbarkeit nur die erzielte Genauigkeit mit durchschnittlich 58% bis 88% an. Diese Werte sind aber wie in Abschnitt 4.3.2 erläutert wenig aussagekräftig, auch weil sie einzelne Kurzschlüsse und nicht Sektoren mit Kurzschlüssen zählten.[76]

Als Chan et al. einen modernen Entwurf für eine 'sub 14nm' Prozesstechnologie untersuchten schaffte ihr Vorhersagemodell ein Richtig-Positiv Rate von 74% bei einer sehr geringen Falsch-Positiv Rate von 0,2%. Gerade ihre Falsch-Positiv Rate ist deutlich niedriger als die in dieser Arbeit erreichte.[10]

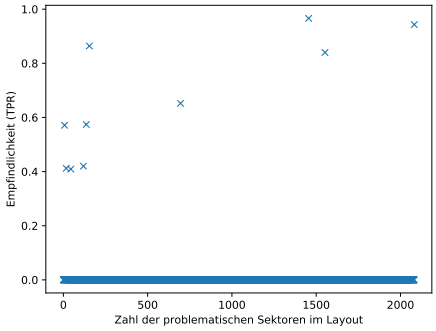


(a) Specificity

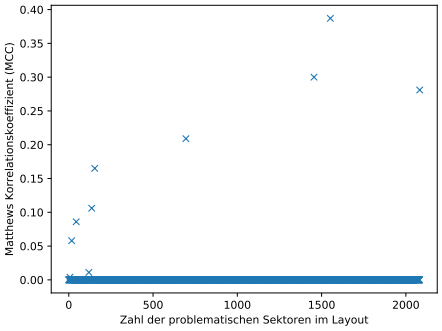


(b) Genauigkeit (Anteil der wahren Vorhersagen an der Gesamtheit)

Abbildung 5.2: Negative Korrelation zwischen Metriken für die Vorhersage von problemfreien Stellen und Anzahl der Kurzschlüsse pro Layout



(a) Sensitivität (Wahrpositivrate)



(b) Matthews Korrelationskoeffizient mit einem Wertebereich von $[-1, +1]$ wobei $+1$ der beste Wert ist.

Abbildung 5.3: Positive Korrelation zwischen Metriken für die Vorhersage von problemfreien Stellen und Anzahl der Kurzschlüsse pro Layout

6 Zusammenfassung

6.1 Fazit

Das Ziel dieser Arbeit, anhand platzierter Layouts Fehler in verdrahteten Layouts vorherzusagen, bedurfte vieler Vorbereitungen bevor überhaupt mit der Entwicklung des Vorhersage Algorithmus begonnen werden konnte.

Software Werkzeuge und Datenmaterial für Elektronik Entwurfsautomatisierung besaßen im reinen Opensource Bereich nicht die Fähigkeit mit der Komplexität moderner Digitalschaltungen umzugehen, welche die Vorhersage von Verdrahtungsfehlern erst möglich macht. Im akademischen Closed Source Bereich waren geeignete Benchmarks (von den ISPD Wettbewerben 2014/2015) und Platzierer (Eh?Placer) vorhanden, es scheiterte jedoch an der Detailverdrahtung. Dafür musste ein kommerzielles Werkzeug eingesetzt werden, die Encounter Digital Implementation Software. Zur Extraktion von relevanten Features für ein maschinelles Lernen Modell wurden entsprechende Daten aus der Encounter Datenbank für jedes Layout abgerufen.

Der ursprüngliche Plan, die Verdrahtbarkeit aus Bildern der Sektoren abzuleiten, wurde schlussendlich verworfen, da diese allein nicht die notwendigen Informationen über Netze zwischen weit entfernten Gitterzellen beinhalten. Zudem hätte die Verarbeitung von Bilddaten deutlich größere Rechenressourcen benötigt als die in dieser Arbeit erfassten acht Zahlenwerte pro Sektor.

Die Optimierung des verwendeten neuronalen Netzes geschah überwiegend heuristisch. Einzelne Hyperparameter wie die geringe Anzahl der Schichten konnten aufgrund der wenig komplexen Eingangsdaten annähernd geraten werden. Die optimale Zahl der Trainingsepochen wurde experimentell bestimmt.

Das trainierte Modell traf Vorhersagen über das Vorhandensein von Kurzschlüssen mit deutlich größerer Genauigkeit als bei zufälligem Raten zu erwarten wäre. Es blieb aber hinter den Genauigkeiten zurück, die von anderen Forschern erzielt werden konnten. Angesichts des großen Rechenaufwands für die Detailverdrahtung können die Ergebnisse dieser Arbeit von Nutzen sein um iterativ eine Platzierung zu verbessern.

6.2 Ausblick

Detailliertere Eingabedaten und Vorhersagen

Bisher wurden nur aus den platzierten Layoutdaten extrahierte Eigenschaften (Features), wie zum Beispiel die Anzahl der lokalen und globalen Netze in einer Gitterzelle, verwendet um die Verdrahtbarkeit vorherzusagen. Zur Vorhersage von lithografischen Hotspots waren Bilder die Eingangsdaten. Interessant wäre es die Bilder der betreffenden Gitterzellen ebenso wie die extrahierten Eigenschaften in einem kombinierten Modell auszuwerten. Dieses würde für die Bilder-

kennung ein *Convolutional Neural Network* einsetzen um die visuellen Daten zu abstrahieren. In Kombination mit den Features könnte damit die Verdrahtbarkeit noch genauer bestimmt werden. In dieser Arbeit wurde nur betrachtet, ob in einer Zelle Kurzschlüsse auftreten oder nicht. Mehr Aufschluss könnte die Anzahl der Kurzschlüsse oder der Kurzschlüsse verursachenden Netze geben. Es könnten außerdem getestet werden wie mit zusätzlichen oder anderen Features (siehe Abschnitt 4.3.1) die Vorhersagekraft des Modells gesteigert werden kann.

Verbesserte Hyperparameter des neuronalen Netzes

Für diese Arbeit wurde beim Optimieren der Hyperparameter vor allem die Zahl der Iterationen (Epochen) und Größe der Chargen (Batches) variiert. Mit mehr Rechenleistung könnten zusätzlich Kombinationen aus verwendetem Optimierungsalgorithmus, Verlustfunktion und Zahl, Größe und Aktivierungsfunktion der verborgenen Schichten ausprobiert werden. Die Talos Software[59] hilft bei der Automatisierung dieses Vorgangs. Möglicherweise kann die Verwendung einer Tensor Processing Unit in Google Colaboratory (siehe Abschnitt 4.3.3) gegenüber der Verwendung einer Grafikkarte die notwendige (kostenfreie) Rechenleistung bereitstellen.

Generative Adversarial Learning

Die in dieser Arbeit erwähnten und verwendeten maschinelles Lernen Verfahren benötigen eine große Menge an Trainingsdaten, um eine gute Genauigkeit zu erreichen. Dies limitiert die Entwicklung und Optimierung von Verfahren der Layout-Verifikation mithilfe von maschinellem Lernen, da es gerade in der akademischen Forschung wenig Zugang zu realen Schaltungsentwürfen für moderne Prozesstechnologien gibt.

Neuartige Verfahren wie *Generative Adversarial Networks* (GAN) ermöglichen das Trainieren von neuronalen Netzen mit nur wenig Eingangsdaten. Ein Netz ist ein *generator*, kreiert also neue Daten. Das andere Netz ist ein *discriminator*, versucht also die vom *generator* Netzwerk kreierten Bilder von realen Eingangsdaten zu unterscheiden. Durch den Input vom *discriminator* Netzwerk lernt das *generator* Netzwerk den realen Eingangsdaten täuschend ähnliche Daten zu erzeugen, ohne jemals selbst die originalen Eingangsdaten gesehen zu haben. Zum Beispiel erstellt das eine Netz Bilder, während das andere Netz lernt diese Fälschungen von echten Kunstwerken zu unterscheiden.[12, S.305ff]

Es ist zu untersuchen, ob es möglich ist mit GAN neue Layoutdaten zu erstellen, die Industrielayouts realistisch nachahmen. Diese Layoutdaten könnten die bereits vorhandenen Trainingsdaten erweitern. Dies würde die Vorhersagekraft existierender Modelle erhöhen, auch für so noch nicht real vorgekommene Fehler. Eine Herausforderung dabei ist die erhebliche Rechenleistung, die zur Platzierung und Verdrahtung einer digitalen Schaltung notwendig ist. Je nach Implementierung müsste das bei jedem neu erstellten Layout des Generator Netzwerks geschehen.

Zeitschriftenaufsätze

[6] Pieter-Tjerk de Boer, Dirk P. Kroese, Shie Mannor und Reuven Y. Rubinstein. „A Tutorial on the Cross-Entropy Method“. In: *Annals of Operations Research* 134.1 (Feb. 2005), S. 19–67.

[11] Davide Chicco. „Ten quick tips for machine learning in computational biology“. In: *BioData Mining* 10.1 (Dez. 2017).

[18] Timothy Dozat. „Incorporating nesterov momentum into adam“. In: (2016).

[29] Jinwook Jung u. a. „DATC RDF: An Open Design Flow from Logic Synthesis to Detailed Routing“. In: *arXiv:1810.01078 [cs]* (2. Okt. 2018).

[32] Yoon Kim. „Convolutional Neural Networks for Sentence Classification“. In: (25. Aug. 2014).

[33] Diederik P. Kingma und Jimmy Ba. „Adam: A Method for Stochastic Optimization“. In: (22. Dez. 2014).

[42] Warren S. McCulloch und Walter Pitts. „A logical calculus of the ideas immanent in nervous activity“. In: *The Bulletin of Mathematical Biophysics* 5.4 (Dez. 1943), S. 115–133.

[43] A. Tabrizi et al. N. Karimpour Darav A. Kennings. „Eh?Placer: A High-Performance Modern Technology-Driven Placer“. In: *ACM TO-DAES* 21.3 (2016), 37:1–37:27.

[53] F. Rosenblatt. „The perceptron: A probabilistic model for information storage and organization in the brain.“ In: *Psychological Review* 65.6 (1958), S. 386–408.

[55] Anuj Sharma und Prabin Kumar Panigrahi. „A Review of Financial Accounting Fraud Detection based on Data Mining Techniques“. In: (16. Sep. 2013).

[57] Karen Simonyan und Andrew Zisserman. „Very Deep Convolutional Networks for Large-Scale Image Recognition“. In: (4. Sep. 2014).

[66] Y. Du et al. W.-T. J. Chan. „BEOL Stack-Aware Routability Prediction from Placement Using Data Mining Techniques“. In: *, Proc. ICCD* (2016), S. 41–48.

Aufsätze in Tagungsbänden (Proceedings)

[5] Charles J. Alpert u. a. „What makes a design difficult to route“. In: *Proceedings of the 19th international symposium on Physical design - ISPD 10*. ACM Press, 2010.

[7] Ismail S. Bustany, David Chinnery, Joseph R. Shinnerl und Vladimir Yutsis. „ISPD 2015 Benchmarks with Fence Regions and Routing Blockages for Detailed-Routing-Driven Placement“. In: *Proceedings of the 2015 Symposium on International Symposium on Physical Design - ISPD '15*. the 2015 Symposium. Monterey, California, USA: ACM Press, 2015, S. 157–164.

[10] Wei-Ting J. Chan, Pei-Hsin Ho, Andrew B. Kahng und Prashant Saxena. „Routability Optimization for Industrial Designs at Sub-14Nm Process Nodes Using Machine Learning“. In: *Proceedings of the 2017 ACM on International Symposium on Physical Design*. ISPD '17. event-place: Portland, Oregon, USA. New York, NY, USA: ACM, 2017, S. 15–21.

[30] Andrew B. Kahng, Hyein Lee und Jiajia Li. „Horizontal benchmark extension for improved assessment of physical CAD research“. In: *Proceedings of the 24th edition of the great lakes symposium on VLSI*. ACM. 2014, S. 27–32.

[56] Hamid Shojaei, Azadeh Davoodi und Jeffrey T Linderoth. „Congestion analysis for global routing via integer programming“. In: *Proceedings of the International Conference on Computer-Aided Design*. IEEE Press. 2011, S. 256–262.

[58] Aysa Fakheri Tabrizi u. a. „A machine learning framework to identify detailed routing short violations from a placed netlist“. In: *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 2018, S. 1–6.

[64] J Torres. „ICCAD-2012 CAD contest in fuzzy pattern matching for physical verification and benchmark suite“. In: Nov. 2012, S. 349–350.

[73] Haoyu Yang u. a. „Layout Hotspot Detection with Feature Tensor Generation and Deep Biased Learning“. In: *Proceedings of the 54th Annual Design Automation Conference 2017 on - DAC '17*. the 54th Annual Design Automation Conference 2017. Austin, TX, USA: ACM Press, 2017, S. 1–6.

[74] Meng-Lin Yu. „A Study of the Applicability of Hopfield Decision Neural Nets to VLSI CAD“. In: *26th ACM/IEEE Design Automation Conference*. Juni 1989, S. 412–417.

[75] Hang Zhang, Bei Yu und Evangeline F. Y. Young. „Enabling online learning in lithography hotspot detection with information-theoretic feature optimization“. In: *Proceedings of the 35th International Conference on Computer-Aided Design - ICCAD 16*. ACM Press, 2016.

[76] Quan Zhou u. a. „An accurate detailed routing routability prediction model in placement“. In: *2015 6th Asia Symposium on Quality Electronic Design (ASQED)*. 2015 6th Asia Symposium on Quality Electronic Design (ASQED). Kula Lumpur, Malaysia: IEEE, Aug. 2015, S. 119–122.

Patente

[3] Jing Su et al. „Identification of hot spots or defects by machine learning“. US20190147127A1. ASML Netherlands B.V. 16. Mai 2019.

[14] Yi-Lin Chuang u. a. „Maschinenlernende Entwurfsplattform“. US-Pat. DE102017123446A1. Ltd. Taiwan Semiconductor Manufacturing Co. 2018.

[31] Manadher Kharroubi. „System and method for designing a chip floorplan using machine learning“. US20190205494A1. Inc. Arteris. 2018.

[52] Juan Andres Torres Robles, Salma Mostafa Fahmy, Kareem Madkour und Jen-Yi Wuu. „Hotspot detection based on machine learning“. US8402397B2. Mentor Graphics Corp. 2011.

[68] Samuel I. Ward. „Machine-Learning based datapath extraction“. US9147032B2. Global Foundries Inc. 2015.

[69] Alan J. Leslie Wei Guo. „Automated lithographic hot spot detection employing unsupervised topological image categorization“. US8453075B2. 2. Sep. 2011.

Bücher

[12] Francois Chollet. *Deep Learning with Python.* Manning Publications, 28. Okt. 2017. 384 S.

[25] D. O. Hebb. *The Organization of Behavior.* Taylor & Francis Ltd., 11. Apr. 2005.

[37] S. Y. Kung. *Kernel Methods and Machine Learning.* Cambridge University Press, 2014.

[40] Jens Lienig. *Layoutsynthese elektronischer Schaltungen.* Springer Berlin Heidelberg, 2016.

[65] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory.* Springer New York, 1995.

Internet

[1] Martıén Abadi u. a. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[2] *About LibreCores*. URL: https://www.librecores.org/static/about (besucht am 26. 08. 2019).

[4] Alisneaky. *Kernel Machine*. URL: https://commons.wikimedia.org/wiki/File:Kernel_Machine.png (besucht am 22. 08. 2019). Lizenzfrei: https://creativecommons.org/publicdomain/zero/1.

[8] Oliscience BV. *Opencores. open logic interconnects science*. URL: https://oliscience.nl/#OpenCores (besucht am 23. 08. 2019).

[9] Cecbur. *A filter in a convolutional artificial neural network. A filter (=kernel, neuron) in a convolutional artificial neural network. The input to the filter is three features thick. The three features come from three separate filters in the previous layer of the deep neural network*. modifiziert. 23. Aug. 2019. URL: https://commons.wikimedia.org/wiki/File:Convolutional_Neural_Network_NeuralNetworkFeatureLayers.gif (besucht am 23. 08. 2019). Lizenz: https://creativecommons.org/licenses/by-sa/4.0/deed.de.

[13] François Chollet u. a. *Keras*. 2015.

[15] *Cloud Tensor Processing Units (TPUs)*. Google Cloud. URL: https://cloud.google.com/tpu/docs/tpus (besucht am 03. 09. 2019).

[16] Mysid Dake. *Neural network*. URL: https://commons.wikimedia.org/wiki/File:Neural_network.svg (besucht am 23. 08. 2019). Lizenz: https://creativecommons.org/licenses/by/1.0/

[17] Nima Karimpour Darav. *Eh?Placer: A High-Performance Modern Technology-Driven Placer*. URL: https://www.ucalgary.ca/karimpour/node/10 (besucht am 24. 08. 2019).

[19] R. Timothy Edwards. *Open Circuit Design. bright ideas. . . no strings attached*. URL: http://opencircuitdesign.com/ (besucht am 24. 08. 2019).

[20] R. Timothy Edwards. *Qrouter 1.3 (Stable) and 1.4 (Development). Qrouter version 1.3 (stable) and 1.4 (development) multi-level, over-the-cell maze router*. URL: http://opencircuitdesign.com/~tim/programs/qrouter/index.html (besucht am 24. 08. 2019).

[21] R. Timothy Edwards. *R. Timothy Edwards. Github Repository*. Open Circuit Design. URL: https://github.com/RTimothyEdwards (besucht am 24. 08. 2019).

[22] Olivier Girard. *openMSP430 :: Overview*. URL: https://opencores.org/projects/openmsp430 (besucht am 23. 08. 2019).

[23] A. Davoodi H. Shojaei und J. Linderoth. *CGRIP: Global Routing Congestion Analysis for Modern VLSI Design*. URL: http://homepages.cae.wisc.edu/~adavoodi/gr/cgrip.htm (besucht am 24. 08. 2019).

[24] Yosys Headquarters. *nextpnr – a portable FPGA place and route tool.* URL: https://github.com/YosysHQ/nextpnr (besucht am 24. 08. 2019).

[26] Richard Herveille. *I2C controller core :: Overview.* URL: https://opencores.org/projects/i2c (besucht am 23. 08. 2019).

[27] *International Symposium on Physical Design.* P. URL: http://www.ispd.cc/?page=contests (besucht am 26. 08. 2019).

[28] *ISPD 2012 Discrete Gate Sizing Contest and Benchmark Suite.* URL: http://www.ispd.cc/contests/12/ispd2012_contest.html (besucht am 26. 08. 2019).

[34] Matthias Köfferlein. *About The Project. KLayout Highlights.* URL: https://www.klayout.de/intro.html (besucht am 26. 08. 2019).

[35] Matthias Köfferlein. *KLayout - High Performance Layout Viewer And Editor.* URL: https://www.klayout.de/ (besucht am 26. 08. 2019).

[38] Larhmam. *SVM margin. Maximum-margin hyperplane and margin for an SVM trained on two classes. Samples on margins are called support vectors.* URL: https://en.wikipedia.org/wiki/File:SVM_margin.png (besucht am 22. 08. 2019). Lizenz: https://creativecommons.org/licenses/by-sa/4.0/legalcode.

[39] *LibreCores. We drive Free and Open Digital Hardware.* The Free und Open Source Silicon Foundation. URL: https://www.librecores.org/ (besucht am 26. 08. 2019).

[41] Wen-Hao Liu, Ke-Ren Dai, Wei-Chun Kao und Yih-Lang Li. *NCTU-GR 2.0.* URL: https://people.cs.nctu.edu.tw/~whliu/NCTU-GR.htm (besucht am 25. 08. 2019).

[44] Stefan Nolting. *neo430. by zerogravity.* URL: https://www.librecores.org/zerogravity/neo430.

[45] equivalent to Oliscience BV OpenCores.org. *OpenCores.* URL: https://opencores.org/ (besucht am 23. 08. 2019).

[46] Oritnk. *binary confusion matrix.* URL: https://en.wikipedia.org/wiki/File:Binary_confusion_matrix.png (besucht am 09. 09. 2019).

[47] *Overview of JPEG.* Joint Photographic Experts Group (JPEG) committee. URL: https://jpeg.org/jpeg/ (besucht am 17. 08. 2019).

[48] International Symposium on Physical Design. *ISPD 2014 Detailed Routing-Driven Placement Contest.* URL: http://www.ispd.cc/contests/14/ispd2014_contest.html (besucht am 10. 07. 2019).

[49] International Symposium on Physical Design. *ISPD 2015 Blockage-Aware Detailed Routing-Driven Placement Contest.* URL: http://www.ispd.cc/contests/14/ispd2015_contest.html (besucht am 10. 07. 2019).

[50] *Proton : ASIC Place and Route Suite.* Efabless. URL: https://github.com/efabless/proton (besucht am 25. 08. 2019).

[51] *Qflow 1.3: An Open-Source Digital Synthesis Flow.* URL: http://opencircuitdesign.com/qflow/ (besucht am 24. 08. 2019).

[54] rubund. URL: https://github.com/rubund/graywolf (besucht am 24. 08. 2019).

[59] Autonomio Talos. *Hyperparameter Optimization for Keras Models.* 2019. URL: https://github.com/autonomio/talos (besucht am 22. 08. 2019).

[60] *TAU Contest 2017. Contest on timing macro-modeling.* 2017. URL: https://sites.google.com/site/taucontest2017/ (besucht am 04. 09. 2019).

[61] *TF 2.0.0rc0 Cannot connect to TPU device - Issue 32043.* Google. URL: https://github.com/tensorflow/tensorflow/issues/32043 (besucht am 03. 09. 2019).

[62] *The Free and Open Source Silicon Foundation.* URL: https://www.fossi-foundation.org/ (besucht am 26. 08. 2019).

[63] Mohamed Kassem Tim Edwards. *The Raven chip. First-time silicon success with qflow and efabless.* efabless. URL: https://s3-us-west-1.amazonaws.com/efabless-production-marketplace/assets/Qflow_Raven_FSiC2019.pdf (besucht am 24. 08. 2019).

[67] Matthias von Wachter. *Routability Prediction with Machine Learning.* URL: https://gitlab.hrz.tu-chemnitz.de/IFTE-EDA/rpml (besucht am 03. 09. 2019).

[70] *Welcome to Colaboratory!* Institut für Feinwerktechnik und Elektronikdesign an der TU Dresden. URL: https://colab.research.google.com/notebooks/welcome.ipynb (besucht am 03. 09. 2019).

[71] Clifford Wolf. *Yosys Open SYnthesis Suite.*

[72] Clifford Wolf und Mathias Lasser. *Project IceStorm.*

Literatur

[1] Martıén Abadi u. a. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[2] *About LibreCores*. URL: `https://www.librecores.org/static/about` (besucht am 26. 08. 2019).

[3] Jing Su et al. „Identification of hot spots or defects by machine learning“. US20190147127A1. ASML Netherlands B.V. 16. Mai 2019.

[4] Alisneaky. *Kernel Machine*. URL: `https://commons.wikimedia.org/wiki/File:Kernel_Machine.png` (besucht am 22. 08. 2019). Lizenzfrei: https://creativecommons.org/publicdomain/zero/1.

[5] Charles J. Alpert u. a. „What makes a design difficult to route“. In: *Proceedings of the 19th international symposium on Physical design - ISPD 10*. ACM Press, 2010.

[6] Pieter-Tjerk de Boer, Dirk P. Kroese, Shie Mannor und Reuven Y. Rubinstein. „A Tutorial on the Cross-Entropy Method“. In: *Annals of Operations Research* 134.1 (Feb. 2005), S. 19–67.

[7] Ismail S. Bustany, David Chinnery, Joseph R. Shinnerl und Vladimir Yutsis. „ISPD 2015 Benchmarks with Fence Regions and Routing Blockages for Detailed-Routing-Driven Placement“. In: *Proceedings of the 2015 Symposium on International Symposium on Physical Design - ISPD '15*. the 2015 Symposium. Monterey, California, USA: ACM Press, 2015, S. 157–164.

[8] Oliscience BV. *Opencores. open logic interconnects science*. URL: `https://oliscience.nl/#OpenCores` (besucht am 23. 08. 2019).

[9] Cecbur. *A filter in a convolutional artificial neural network. A filter (=kernel, neuron) in a convolutional artificial neural network. The input to the filter is three features thick. The three features come from three separate filters in the previous layer of the deep neural network.* modifiziert. 23. Aug. 2019. URL: `https://commons.wikimedia.org/wiki/File:Convolutional_Neural_Network_NeuralNetworkFeatureLayers.gif` (besucht am 23. 08. 2019). Lizenz: https://creativecommons.org/licenses/by-sa/4.0/deed.de.

[10] Wei-Ting J. Chan, Pei-Hsin Ho, Andrew B. Kahng und Prashant Saxena. „Routability Optimization for Industrial Designs at Sub-14Nm Process Nodes Using Machine Learning“. In: *Proceedings of the 2017 ACM on International Symposium on Physical Design*. ISPD '17. event-place: Portland, Oregon, USA. New York, NY, USA: ACM, 2017, S. 15–21.

[11] Davide Chicco. „Ten quick tips for machine learning in computational biology“. In: *BioData Mining* 10.1 (Dez. 2017).

[12] Francois Chollet. *Deep Learning with Python*. Manning Publications, 28. Okt. 2017. 384 S.

[13] François Chollet u. a. *Keras*. 2015.

[14] Yi-Lin Chuang u. a. „Maschinenlernende Entwurfsplattform“. US-Pat. DE102017123446A1. Ltd. Taiwan Semiconductor Manufacturing Co. 2018.

[15] *Cloud Tensor Processing Units (TPUs)*. Google Cloud. URL: `https://cloud.google.com/tpu/docs/tpus` (besucht am 03. 09. 2019).

[16] Mysid Dake. *Neural network*. URL: `https://commons.wikimedia.org/wiki/File:Neural_network.svg` (besucht am 23. 08. 2019). Lizenz: https://creativecommons.org/licenses/by/1.0/

[17] Nima Karimpour Darav. *Eh?Placer: A High-Performance Modern Technology-Driven Placer*. URL: `https://www.ucalgary.ca/karimpour/node/10` (besucht am 24. 08. 2019).

[18] Timothy Dozat. „Incorporating nesterov momentum into adam“. In: (2016).

[19] R. Timothy Edwards. *Open Circuit Design. bright ideas. . . no strings attached*. URL: `http://opencircuitdesign.com/` (besucht am 24. 08. 2019).

[20] R. Timothy Edwards. *Qrouter 1.3 (Stable) and 1.4 (Development). Qrouter version 1.3 (stable) and 1.4 (development) multi-level, over-the-cell maze router*. URL: `http://opencircuitdesign.com/~tim/programs/qrouter/index.html` (besucht am 24. 08. 2019).

[21] R. Timothy Edwards. *R. Timothy Edwards. Github Repository*. Open Circuit Design. URL: `https://github.com/RTimothyEdwards` (besucht am 24. 08. 2019).

[22] Olivier Girard. *openMSP430 :: Overview*. URL: `https://opencores.org/projects/openmsp430` (besucht am 23. 08. 2019).

[23] A. Davoodi H. Shojaei und J. Linderoth. *CGRIP: Global Routing Congestion Analysis for Modern VLSI Design*. URL: `http://homepages.cae.wisc.edu/~adavoodi/gr/cgrip.htm` (besucht am 24. 08. 2019).

[24] Yosys Headquarters. *nextpnr – a portable FPGA place and route tool*. URL: `https://github.com/YosysHQ/nextpnr` (besucht am 24. 08. 2019).

[25] D. O. Hebb. *The Organization of Behavior*. Taylor & Francis Ltd., 11. Apr. 2005.

[26] Richard Herveille. *I2C controller core :: Overview*. URL: `https://opencores.org/projects/i2c` (besucht am 23. 08. 2019).

[27] *International Symposium on Physical Design*. P. URL: `http://www.ispd.cc/?page=contests` (besucht am 26. 08. 2019).

[28] *ISPD 2012 Discrete Gate Sizing Contest and Benchmark Suite*. URL: `http://www.ispd.cc/contests/12/ispd2012_contest.html` (besucht am 26. 08. 2019).

[29] Jinwook Jung u. a. „DATC RDF: An Open Design Flow from Logic Synthesis to Detailed Routing“. In: *arXiv:1810.01078 [cs]* (2. Okt. 2018).

[30] Andrew B. Kahng, Hyein Lee und Jiajia Li. „Horizontal benchmark extension for improved assessment of physical CAD research“. In: *Proceedings of the 24th edition of the great lakes symposium on VLSI*. ACM. 2014, S. 27–32.

[31] Manadher Kharroubi. „System and method for designing a chip floorplan using machine learning“. US20190205494A1. Inc. Arteris. 2018.

[32] Yoon Kim. „Convolutional Neural Networks for Sentence Classification“. In: (25. Aug. 2014).

[33] Diederik P. Kingma und Jimmy Ba. „Adam: A Method for Stochastic Optimization“. In: (22. Dez. 2014).

[34] Matthias Köfferlein. *About The Project. KLayout Highlights*. URL: `https://www.klayout.de/intro.html` (besucht am 26. 08. 2019).

[35] Matthias Köfferlein. *KLayout - High Performance Layout Viewer And Editor*. URL: https://www.klayout.de/ (besucht am 26. 08. 2019).

[36] Alex Krizhevsky, Ilya Sutskever und Geoffrey E Hinton. „ImageNet Classification with Deep Convolutional Neural Networks“. In: *Advances in Neural Information Processing Systems 25*. Hrsg. von F. Pereira, C. J. C. Burges, L. Bottou und K. Q. Weinberger. Curran Associates, Inc., 2012, S. 1097–1105.

[37] S. Y. Kung. *Kernel Methods and Machine Learning*. Cambridge University Press, 2014.

[38] Larhmam. *SVM margin. Maximum-margin hyperplane and margin for an SVM trained on two classes. Samples on margins are called support vectors*. URL: https://en.wikipedia.org/wiki/File:SVM_margin.png (besucht am 22. 08. 2019). Lizenz: https://creativecommons.org/licenses/by-sa/4.0/legalcode.

[39] *LibreCores. We drive Free and Open Digital Hardware*. The Free und Open Source Silicon Foundation. URL: https://www.librecores.org/ (besucht am 26. 08. 2019).

[40] Jens Lienig. *Layoutsynthese elektronischer Schaltungen*. Springer Berlin Heidelberg, 2016.

[41] Wen-Hao Liu, Ke-Ren Dai, Wei-Chun Kao und Yih-Lang Li. *NCTU-GR 2.0*. URL: https://people.cs.nctu.edu.tw/~whliu/NCTU-GR.htm (besucht am 25. 08. 2019).

[42] Warren S. McCulloch und Walter Pitts. „A logical calculus of the ideas immanent in nervous activity“. In: *The Bulletin of Mathematical Biophysics* 5.4 (Dez. 1943), S. 115–133.

[43] A. Tabrizi et al. N. Karimpour Darav A. Kennings. „Eh?Placer: A High-Performance Modern Technology-Driven Placer“. In: *ACM TO-DAES* 21.3 (2016), 37:1–37:27.

[44] Stefan Nolting. *neo430. by zerogravity*. URL: https://www.librecores.org/zerogravity/neo430.

[45] equivalent to Oliscience BV OpenCores.org. *OpenCores*. URL: https://opencores.org/ (besucht am 23. 08. 2019).

[46] Oritnk. *binary confusion matrix*. URL: https://en.wikipedia.org/wiki/File:Binary_confusion_matrix.png (besucht am 09. 09. 2019).

[47] *Overview of JPEG*. Joint Photographic Experts Group (JPEG) committee. URL: https://jpeg.org/jpeg/ (besucht am 17. 08. 2019).

[48] International Symposium on Physical Design. *ISPD 2014 Detailed Routing-Driven Placement Contest*. URL: http://www.ispd.cc/contests/14/ispd2014_contest.html (besucht am 10. 07. 2019).

[49] International Symposium on Physical Design. *ISPD 2015 Blockage-Aware Detailed Routing-Driven Placement Contest*. URL: http://www.ispd.cc/contests/14/ispd2015_contest.html (besucht am 10. 07. 2019).

[50] *Proton : ASIC Place and Route Suite*. Efabless. URL: https://github.com/efabless/proton (besucht am 25. 08. 2019).

[51] *Qflow 1.3: An Open-Source Digital Synthesis Flow*. URL: http://opencircuitdesign.com/qflow/ (besucht am 24. 08. 2019).

[52] Juan Andres Torres Robles, Salma Mostafa Fahmy, Kareem Madkour und Jen-Yi Wuu. „Hotspot detection based on machine learning“. US8402397B2. Mentor Graphics Corp. 2011.

[53] F. Rosenblatt. „The perceptron: A probabilistic model for information storage and organization in the brain.“ In: *Psychological Review* 65.6 (1958), S. 386–408.

[54] rubund. URL: https://github.com/rubund/graywolf (besucht am 24. 08. 2019).

[55] Anuj Sharma und Prabin Kumar Panigrahi. „A Review of Financial Accounting Fraud Detection based on Data Mining Techniques“. In: (16. Sep. 2013).

[56] Hamid Shojaei, Azadeh Davoodi und Jeffrey T Linderoth. „Congestion analysis for global routing via integer programming“. In: *Proceedings of the International Conference on Computer-Aided Design*. IEEE Press. 2011, S. 256–262.

[57] Karen Simonyan und Andrew Zisserman. „Very Deep Convolutional Networks for Large-Scale Image Recognition“. In: (4. Sep. 2014).

[58] Aysa Fakheri Tabrizi u. a. „A machine learning framework to identify detailed routing short violations from a placed netlist“. In: *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 2018, S. 1–6.

[59] Autonomio Talos. *Hyperparameter Optimization for Keras Models*. 2019. URL: `https://github.com/autonomio/talos` (besucht am 22. 08. 2019).

[60] *TAU Contest 2017. Contest on timing macro-modeling*. 2017. URL: `https://sites.google.com/site/taucontest2017/` (besucht am 04. 09. 2019).

[61] *TF 2.0.0rc0 Cannot connect to TPU device - Issue 32043*. Google. URL: `https://github.com/tensorflow/tensorflow/issues/32043` (besucht am 03. 09. 2019).

[62] *The Free and Open Source Silicon Foundation*. URL: `https://www.fossi-foundation.org/` (besucht am 26. 08. 2019).

[63] Mohamed Kassem Tim Edwards. *The Raven chip. First-time silicon success with qflow and efabless*. efabless. URL: `https://s3-us-west-1.amazonaws.com/efabless-production-marketplace/assets/Qflow_Raven_FSiC2019.pdf` (besucht am 24. 08. 2019).

[64] J Torres. „ICCAD-2012 CAD contest in fuzzy pattern matching for physical verification and benchmark suite“. In: Nov. 2012, S. 349–350.

[65] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer New York, 1995.

[66] Y. Du et al. W.-T. J. Chan. „BEOL Stack-Aware Routability Prediction from Placement Using Data Mining Techniques“. In: *, Proc. ICCD* (2016), S. 41–48.

[67] Matthias von Wachter. *Routability Prediction with Machine Learning*. URL: `https://gitlab.hrz.tu-chemnitz.de/IFTE-EDA/rpml` (besucht am 03. 09. 2019).

[68] Samuel I. Ward. „Machine-Learning based datapath extraction“. US9147032B2. Global Foundries Inc. 2015.

[69] Alan J. Leslie Wei Guo. „Automated lithographic hot spot detection employing unsupervised topological image categorization“. US8453075B2. 2. Sep. 2011.

[70] *Welcome to Colaboratory!* Institut für Feinwerktechnik und Elektronikdesign an der TU Dresden. URL: `https://colab.research.google.com/notebooks/welcome.ipynb` (besucht am 03. 09. 2019).

[71] Clifford Wolf. *Yosys Open SYnthesis Suite*.

[72] Clifford Wolf und Mathias Lasser. *Project IceStorm*.

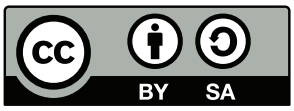
[73] Haoyu Yang u. a. „Layout Hotspot Detection with Feature Tensor Generation and Deep Biased Learning“. In: *Proceedings of the 54th Annual Design Automation Conference 2017 on - DAC '17*. the 54th Annual Design Automation Conference 2017. Austin, TX, USA: ACM Press, 2017, S. 1–6.

[74] Meng-Lin Yu. „A Study of the Applicability of Hopfield Decision Neural Nets to VLSI CAD“. In: *26th ACM/IEEE Design Automation Conference*. Juni 1989, S. 412–417.

[75] Hang Zhang, Bei Yu und Evangeline F. Y. Young. „Enabling online learning in lithography hotspot detection with information-theoretic feature optimization“. In: *Proceedings of the 35th International Conference on Computer-Aided Design - ICCAD 16*. ACM Press, 2016.

[76] Quan Zhou u. a. „An accurate detailed routing routability prediction model in placement“. In: *2015 6th Asia Symposium on Quality Electronic Design (ASQED)*. 2015 6th Asia Symposium on Quality Electronic Design (ASQED). Kula Lumpur, Malaysia: IEEE, Aug. 2015, S. 119–122.

Lizenzbedingungen



Die zu dieser Arbeit gehörenden Skripte stehen in einem Gitlab Repository (`https://gitlab.hrz.tu-chemnitz.de/IFTE-EDA/rpml`) unter der MIT Lizenz zur Verfügung.

Für Zugang kann das Institut für Feinwerktechnik und Elektronikdesign an der Technischen Universität Dresden (`https://www.ifte.de`) kontaktiert werden.