FernUniversität in Hagen
Fakultät für Mathematik und Informatik

Artificial Intelligence Group

Einfluss der KIPF-WELLING-Konvolution auf die Leistung von GNNs mit und ohne lineare Ausleseschichten bei SE_{id}

Bachelorarbeit

zur Erlangung des Grades einer Bachelor of Science (B.Sc.)
im Studiengang Informatik

vorgelegt von
Daniel Boy

Erstgutachter:	Prof. Dr. Matthias Thimm Artificial Intelligence Group
Betreuer:	Prof. Dr. Matthias Thimm Artificial Intelligence Group

Erklärung

Ich erkläre, dass ich die Bachelorarbeit selbstständig und ohne unzulässige Inanspruchnahme Dritter verfasst habe. Ich habe dabei nur die angegebenen Quellen und Hilfsmittel verwendet und die aus diesen wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht. Die Versicherung selbstständiger Arbeit gilt auch für enthaltene Zeichnungen, Skizzen oder graphische Darstellungen. Die Bachelorarbeit wurde bisher in gleicher oder ähnlicher Form weder derselben noch einer anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht. Mit der Abgabe der elektronischen Fassung der endgültigen Version der Bachelorarbeit nehme ich zur Kenntnis, dass diese mit Hilfe eines Plagiatserkennungsdienstes auf enthaltene Plagiate geprüft werden kann und ausschließlich für Prüfungszwecke gespeichert wird.

	Ja	Nein
Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit auf der Webseite des Lehrgebiets Künstliche Intelligenz stimme ich zu.	☐	☐
Der Text dieser Arbeit ist unter einer Creative Commons Lizenz (CC BY-SA 4.0) verfügbar.	☐	☐
Der Quellcode ist unter einer GNU General Public License (GPLv3) verfügbar.	☐	☐
Die erhobenen Daten sind unter einer Creative Commons Lizenz (CC BY-SA 4.0) verfügbar.	☐	☐

.....

(Ort, Datum) (Unterschrift)

Zusammenfassung

In dieser Bachelorarbeit nutzen wir Graphen-basierte Neurale Netzwerke, um abstrakte Argumentationssysteme bezüglich der idealen Semantik approximativ zu lösen. Wir vergleichen dabei ein breites Spektrum an GNN-Designs, die sich in der Normalisierungstechnik und Anzahl der GNN-Schichten unterscheiden, sowie bis zu fünf angehängte lineare Schichten zur Verbesserung der Auslesefunktion enthalten. Wir trainieren und testen die Designs an drei verschiedenen Datensätzen mit unterschiedlichen Eigenschaften. Unsere Ergebnisse zeigen, dass die Einbindung linearer Schichten in ein Design das erfolgreiche Auslesen erheblich verbessert und sich die Leistung unserer erfolgreichsten Designs den höchsten in der bisherigen Forschung berichteten Werten nähert. Diese Einbindung ermöglicht außerdem tiefere GNNs. Wir zeigen schließlich, dass die KIPF-WELLING-Normalisierung für die meisten Typen von abstrakten Argumentationssystemen mit Ausnahme des kwt-Typs einen geringen negativen Einfluss auf die Leistung hat, wenn wir Designs mit und Designs ohne die Normalisierung vergleichen.

Abstract

In this bachelor thesis we use a GNN-based approach to approximately solve abstract argumention frameworks concerning ideal semantics. Specifically we compare a broad range of GNN designs that differ in normalization technique and GNN-layer depth, but also incorporate up to five linear back layers enhancing the read out. We train and test the designs on three contrasting datasets with varying properties. Our results show that the inclusion of linear layers in a design substantially enhances a successful read out and our best designs approach SOTA performance. This inclusion further enables deeper GNNs. Finally we show that the KIPF-WELLING-normalization has a small negative impact on performance for most types of abstract argumention frameworks except the kwt type, when you compare designs with to designs without the normalization.

Inhaltsverzeichnis

1 Einführung

Nicht-monotones Schlussfolgern erlaubt es einem rationalen Agenten, seine Annahmen im Licht neuer Informationen zu revidieren.[1] Eine minimalistische Grundlage dafür bilden abstrakte Argumentationssysteme (kurz AF für engl.: *argumentation framework*). In ihnen wird Argumentation auf Argumente, eine Angriffsrelation zwischen Argumenten, sowie Semantiken heruntergebrochen.[2] Semantiken beschreiben Regeln, wie Argumentenmengen zusammengestellt werden können, so dass sie sinnhafte Positionen widerspiegeln. So unterbinden die Regeln u. a. interne Widersprüche der Position durch das Verbot von Konflikten zwischen Argumenten innerhalb der Position. Leider ist eine korrekte Prüfung auf (Nicht-)Zugehörigkeit eines Argumentes zu Mengen unter bestimmten Semantiken bis zu Π_2^P-schwer,[3] dementsprechend sind übliche Lösungsprogramme ab einer gewissen Größe von abstrakten Argumentationssystemen nicht mehr für den Echtzeiteinsatz nutzbar. Dies macht approximative Verfahren wie den Einsatz maschinellen Lernens wegen ihrer konstanten Laufzeit interessant. Eine Gruppe an Verfahren nutzt Graphenbasierte Neurale Netzwerke (kurz *GNNs* für engl.: *graph neural networks*), die die Struktur eines Graphen für Konvolutionen bzw. Nachrichtenübermittlung nutzen. Dabei hängt die Erkennungsleistung eines trainierten Modells unter anderem von der geschickten Auswahl der Trainingsdaten, einer passenden GNN-Architektur und deren genauer Ausformung ab.[4] Den ersten Versuch maschinelles Lernen auf Graphen mit abstrakter Argumentation zu verbinden starteten 2019 KUHLMANN und THIMM mit einer Machbarkeitsstudie ([KT19]). Weitere Arbeiten in diesem neuen Teilfeld folgten.[5] In dieser Arbeit werden wir die Leistung von verschiedene GNN-Designs bezüglich der bisher nicht behandelten idealen Semantik[6] miteinander vergleichen. Im Gegensatz zu dem in der bisherigen Forschung oft genutzten Problem DS_{pr} ist SE_{id}, die Auflistung aller zur idealen Extension eines AFs gehörigen Argumente, bzw. DC_{id}, ob ein bestimmtes Argument in der idealen Extension liegt, ‚nur' Θ_2^P-schwer.[7] Somit ist ein möglicher Zeitgewinn durch eine approximative Lösung wenn auch kleiner weiterhin groß. Eine weitere Ergänzung unserer Arbeit ist es, dass wir neben Designs mit klassischen GNN-Schichten auch generalisierte Schichten ohne die typische Normierung nutzen. Ebenfalls neu ist, dass wir auf GNN-Schichten auch lineare Schichten folgen lassen, um das Auslesen zu verbessern. Wir untersuchen die Leistung von insgesamt 108 GNN-Designs, die sich

[1] Zur Rolle des nichtmonotonen Schließens in der KI vgl. das ebenso benannte Unterkapitel in [BKI19]: S. 207ff.)

[2] AFs wurden eingeführt von DUNG et al. in [Dun95].

[3] So z. B. das Problem der skeptischen Akzeptanz unter der bevorzugten Semantik: DS_{pr}, vgl. [DBC02].

[4] Wir nennen die allgemeine Beschreibung, wie eine bestimmte Klasse an GNNs Daten verarbeitet, eine GNN-Architektur und einen bestimmten Zusammensetzungplan mit konkreter Schichtenanzahl ein GNN-Design. Wir folgen damit [YYL20].

[5] Für einen Überblick, siehe Abschnitt 3.

[6] Eingeführt in [DMT07].

[7] Vgl. [Dun09].

in Bezug auf Normalisierung, die Anzahl an GNN-Schichten und die Anzahl an linearen Schichten unterscheiden. Dabei prüfen wir die Leistung an drei sehr unterschiedlichen Datensätzen und erzielen Ergebnisse, die an die bisher beste GNN-Architektur AGNN ([CB20]) heranreichen oder sogar übertreffen. Zunächst werden wir in die Theorie einführen. Danach beschreiben und analysieren wir den Stand der Forschung. Im Anschluss betrachten wir die von genutzten Datensätze genauer. Dann beschreiben wir Vortest, unsere Experimentalanordnung und die Ergebnisse unserer Experimente. Abschließend versuchen wir die vorgefundenen Phänomene einzuordnen.

2 Theoretische Grundlagen

In den folgenden Unterkapiteln stellen wir zuerst abstrakte Argumentationssysteme, maschinelles Lernen und Graph-basierte Neuronale Netzwerke vor. Dann führen wir die für den Stand der Forschung und unsere Experimente relevanten Kennzahlen ein. Zuletzt besprechen wir, welche Typen an abstrakten Argumentationssystemen durch welche Generatoren generiert werden.

2.1 Abstrakte Argumentationssysteme

Argumentation ist ein grundlegender Modus, in dem Menschen und andere rationale Agenten Ansichten über sich, über ihre Umgebung und über relevante Konzepte produzieren, erhalten und verändern. „[T]he idea of argumentational reasoning is that a statement is believable if it can be argued successfully against attacking arguments. In other words, whether or not a rational agent believes in a statement depends on whether or not the argument supporting this statement can be successfully defended against the counterarguments. Thus, the beliefs of a rational agent are characterized by the relations between the ‚internal' arguments supporting his beliefs and the ‚external' arguments supporting contradictory beliefs. So, in certain sense, argumentational reasoning is based on the ‚external stability' of the accepted arguments" ([Dun95]: S. 323). Abstrakte Argumentationssysteme sind ein Versuch dieses Konzept generalisiert und verarbeitungsfreundlich abzubilden.[8] Wir definieren Argumentationssysteme (kurz AFs, für engl.: *argumentation framework*) und Extensionen im Folgenden nach [Dun95]. Ein AF ist ein Tupel: $AF = (A, R)$. A ist eine endliche Menge an Argumenten und R eine Teilmenge aus $A \times A$: die Angriffsrelation. AFs sind trivial als gerichtete Graphen mit A als Knoten und R als Kanten darstellbar. Gdw. in einem AF ein Argument $a \in A$ einen Angriff auf ein Argument $b \in A$ darstellt, ist $(a, b) \in R$. Auf Mengen erweitert sagen wir, eine Teilmenge $S \subseteq A$ stellt einen Angriff auf ein Argument $b \in A$ dar, wenn für mindestens ein $a \in S$ ein Angriff $(a, b) \in R$ ist liegt. Alle Argumente, die ein Argument $a \in A$

[8] Alternativen wie DeLP (engl.: *defeasible logic programming*) sind weniger allgemein, dafür expressiver und unterscheiden z. B. über eine Vergleichsrelation schlagende von blockierenden Angriffen (vgl. [BKI19]: S. 317-339).

angreifen, fassen wir in a^- zusammen und alle Argumente, die a selbst angreift in a^+. Analog fassen wir alle Argumente, die mindestens ein Argument einer Menge $S \subseteq A$ angreifen, in S^- zusammen und alle Argumente, die mindestens von einem Argument aus S angegriffen werden, in S^+. Wir sagen eine Teilmenge $S \subseteq A$ verteidigt ein $b \in A$, gdw. es für jeden Angreifer $a \in b^-$ ein Verteidigerargument $v \in S$ gibt, das a wegen $(v, a) \in R$ angreift, also $a \in S^+$ ist. Wir sagen: Das Argument v verteidigt b gegen a.

Wir nennen eine Teilmenge $S \subseteq A$ *konfliktfrei*, wenn kein Argument in S ein anderes in S angreift, also gdw. für kein Paar $a, b \in S$ der Angriff $(a, b) \in R$ ist. Ein Argument $b \in A$ heißt *annehmbar* (engl.: *acceptable*) in Bezug auf die Teilmenge $S \subseteq A$, gdw. $S \cup b$ konfliktfrei ist und auch b verteidigt. Weiterhin nennen wir S dann *zulässig* (engl.: *admissible*), wenn jedes $a \in S$ in Bezug auf S annehmbar ist. S ist dann eine der zulässigen Teilmengen von A: $S \in \text{adm}(AF)$. Eine Teilmenge, die einer bestimmten σ-Semantik (also bestimmten Zusammensetzungsregeln) folgt, nennen wir σ-Extension. Die für unsere Arbeit wichtigsten Extensionen sind die *bevorzugte* und die *ideale* Extension.

S ist eine *bevorzugte* (engl.: *preferred*) Extension E_{pr}, gdw. alle $a \in S$ in Bezug zu S annehmbar sind und jedes $S' \supset S$ nicht mehr konfliktfrei ist. S ist größtmöglich in Bezug auf Mengenvereinigung. S ist dann aus $\text{pr}(AF)$.

S ist die *ideale* (engl.: *ideal*) Extension E_{id}, gdw. alle $a \in S$ in Bezug zu S annehmbar sind, S Teilmenge jeder bevorzugten Extension ist und für jedes $S' \supset S$ gilt,[9] dass kein $a' \in S' \setminus S$ annehmbar in Bezug zu S' ist. S ist dann E_{id} und $S = \text{id}(AF)$.

Ein Nebenrolle spielen die *vollständige*, *grundierte* und *stabile* Extension, die in [CB20] neben der bevorzugten genutzt wurden (siehe unten Abschnitt 3.3).

S ist eine *vollständige* (engl.: *complete*) Extension E_{com}, gdw. jedes $a \in S$ in Bezug zu S annehmbar sind, also $S \in \text{adm}(AF)$ ist, und jedes Argument $b \in A$, das durch S verteidigt wird, selbst in S liegt. S verteidigt kein $c \in A \setminus S$. S ist dann aus $\text{com}(AF)$.

S ist die *grundierte* (engl.: *grounded*) Extension E_{grd}, gdw. S eine vollständige Extension ist und kein $S' \supset S$ eine vollständige Extension ist. S ist dann E_{grd} und $S = \text{grd}(AF)$.

S ist eine *stabile* (engl.: *stable*) Extension E_{stb}, gdw. S konfliktfrei ist und $S \cup S^+ = A$ ist. S greift jedes Argument aus A an, das nicht selbst in S liegt. S ist dann aus $\text{stb}(AF)$.

Unsere Experimente fokussieren sich auf die Akzeptanz von Argumenten bezüglich der idealen Extension eines AFs, ob ein Argument $a \in E_{id}$ ist. Ist dies erfüllt, sagen wir kurz a ist ia, ansonsten ist es ina. Die Menge aller ia Argumente eines AFs nennen wir auch $\text{ia}(AF)$. Wir bezeichnen das Problem der Auflistung aller ia Argumente eines AFs auch kurz mit SE_{id}. Die bisherigen Arbeiten in dem Feld

[9] Dieses Konzept von *ideal* bezeichnen DUNG et al. selbst als „maximal ideal“ (vgl. Theorem 2.1 in [DMT07]: S. 645). Der erste Unterpunkt 2.1.i besagt, dass es für jedes AF genau eine solche Extension gibt: „Every abstract argumentation framework admits a unique maximal ideal set of arguments“ (ebd.).

fokussierten sich vor allem auf die Akzeptanz von Argumenten bezüglich der bevorzugten Extension. Da es mehr als eine bevorzugte Extension per AF geben kann, wird hier *skeptische* (engl.: *sceptical*) von *gutgläubiger* (engl.: *credulous*) Akzeptanz unterschieden. Ein Argument $a \in A$ ist skeptisch akzeptiert bezüglich der bevorzugten Semantik, wenn $\forall E \in \text{prf}(AF) \Rightarrow a \in E$. Ist diese Eigenschaft für ein Argument a erfüllt, sagen wir kurz: a ist pa; a ist ansonsten pna. Die Menge aller pa Argumente eines AFs nennen wir pa(AF). Die Menge aller anderen nennen wir pna(AF). Ein Argument $a \in A$ ist gutgläubig akzeptiert bezüglich der bevorzugten Extension, wenn $\exists E \in \text{prf}(AF) \Rightarrow a \in E$. Ist diese Eigenschaft für ein Argument a erfüllt, sagen wir kurz: a ist pa*; ansonsten ist es pna*. Die Menge aller pa* Argumente eines AFs nennen wir pa*(AF). Die Menge aller anderen nennen wir auch pna*(AF). Jedes ia Argument ist auch auch pa. Jedes pa Argument ist auch auch pa*. Wir bezeichnen das Problem der skeptischen Akzeptanz bezüglich der bevorzugten Semantik kurz mit DS_{pr} und das der gutgäubigen mit DC_{pr}.

2.2 Maschinelles Lernen

Eine andere Spur im Feld der künstlichen Intelligenz führt zum maschinellen Lernen. Statt wie bei abstrakten Argumentationssystemen komplexes Argumentieren in vereinfachte, logische Strukturen zu übertragen werden beim maschinellen Lernen Datenverarbeitungssysteme untersucht, die Lernprozesse beinhalten. Wir sagen, ein solches System lernt „from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E“ ([Mit97]: S. 2). Zur Umsetzung dieses Konzepts gibt es verschiedene Ansätze,[10] wir werden uns auf den Ansatz des tiefen Lernens oder *Deep Learnings* beschränken, der mit Vereinfachungen der biologischen Funktionsweise von Neuronen, Synapsen, ihren eher tiefen Netzen und bestimmten Verarbeitungstechniken Lernprozesse nachbildet. Die nach diesem Ansatz entwickelten Systeme leiten nach einer Trainingsphase aus oft komplexen, neuen und unbekannten Eingaben relevante Informationen für viele Probleme schnell und mit genügender Genauigkeit ab.

2.2.1 Grundlagen des *Deep Learnings*

Im Zentrum des *Deep Learnings* (und anderer Ansätze) stehen in Analogie zu existierenden Neuronentypen entwickelte künstliche Neuronen und ihre Netze. Die Neuronen verarbeiten eingehende Signale und senden Ausgaben weiter (vgl. Abb. 1). Im ersten Verarbeitungsschritt werden die eingehende Signale $h_1 ... h_n$ mit einem Gewicht w pro Eingabe gewichtet und zusammen mit einem Grundwert b (engl.: *bias*) des Neurons über eine Aggregationsfunktion `AGG` wie `SUM` zu einem Zustand s des künstlichen Neurons zusammengefasst. In einem zweiten Schritt wird über

[10] Eine ausführlichere Besprechung, was Lernen im Kontext der Informatik sein kann, finden wir in [BKI19]: S. 99ff.

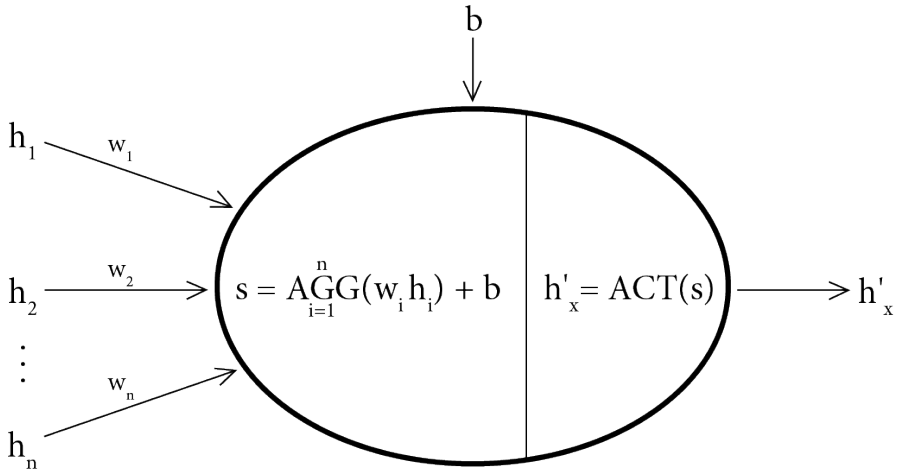

Abbildung 1: Skizze eines einfachen künstlichen Neurons (angepasst aus [KT19]).

eine Aktivierungsfunktion ACT das Ergebnis der Aggregation zu dem ausgehenden Signal h' des künstlichen Neurons verarbeitet.

Das klassische Perzeptron nach ROSENBLATT[11] nutzte wenige eingehende, mit Potentiometern gewichtete Signale, die SUM-Aggregationsfunktion und die HEAVISIDE-Aktivierungsfunktion. Letztere vereinfacht den Entwurf, den Bau und Betrieb mit den Mitteln der 1950er Jahre, erschwert aber als Treppenfunktion die Anpassung der Gewichte in größeren und mehrlagigen Netzen. Diese erste Welle des wissenschaftlichen Interesses an künstlichen neuronalen Netze KNNs (engl.: ANNs für *artificial neural networks*) stieß zügig an ihre Grenzen.[12] RUMELHART, HINTON und WILLIAMS lösten 1986 mit der erfolgreichen Anwendung eines neuen Lernverfahren eine zweite Welle des Interesses aus. Das Fehlerrückführung-Verfahren (engl.: *back-propagation*) ermöglichte größere, eng verknüpfte KNNs.[13] Nach einer Flaute in den 1990er Jahren, erstarkte das Interesse mit tiefen Netzen als *Deep Learning* Mitte der 2000er Jahre erneut.

[11] ROSENBLATT([Ros58]) baute dabei auf theoretischen Vorarbeiten über das biologische Lernen von MCCULLOCH und PITTS([MP43]) auf. Vgl. zu den folgenden Ausführungen zur Geschichte der KNNs [GBC16]: S. 225.

[12] So zeigten MINSKY und PAPERT 1969, dass klassische Perzeptrons keine XOR-Operation durchführen konnten.(vgl. [MP69])

[13] Mehr dazu in [GBC16]: S. 204ff.

Grundsätzlich werden KNNs aus künstlichen Neuronen aufgebaut, die dem Perzeptron ähneln. Dabei werden die künstliche Neuronen zu distinkten Schichten zusammengefasst und differenzierbare Aktivierungsfunktionen wie `Sigmoid` oder `ReLU` genutzt.[14] Die bekannteste KNN-Struktur ist das Mehr-Lagen-Perzeptron, kurz MLP (engl.: *multilayer perceptron*):[15] ein mehrlagiges, vorwärts verarbeitendes KNN (engl.: *feed-forward ANN*). In vorwärts verarbeitenden KNNs sind die Schichten geordnet und Signale bewegen sich nur in aufsteigender Richtung durch die Schichten. Auf eine Eingabeschicht i (auch mit h^0 bezeichnet, obwohl sie nicht versteckt ist) mit n Neuronen folgen m versteckte Schichten h^1 bis h^m und abschließend eine Ausgabeschicht o mit k Neuronen. Die Neuronen der versteckten Schichten haben jeweils einen Zustand s, der durch einen d-dimensionalen Vektor ausgedrückt wird. Wir sagen auch, dass der MLP d Dimensionen oder d versteckte Einheiten (engl.: *hidden units*) hat. Die Kapazität oder Expressivität eines MLPs komplexere Funktionen zu approximieren steigt mit der Zahl der versteckten Neuronen und der Anzahl an Dimensionen ihres Zustandes. Wir nutzen stets 64 Dimensionen für unsere versteckten Schichten. Die Neuronen der Eingabeschicht verarbeiten keine Signale, sondern nehmen die Eingabe des MLPs $X = (x_1, \ldots, x_n)$ verteilt an die Eingangsneuronen auf und senden sie an die Neuronen der ersten versteckten Schicht h_1 weiter.[16] Im Allgemeinen werden die Ausgabesignale eines Neurons an alle Neuronen der Folgeschicht gesendet. In den versteckten Schichten findet der Großteil der Verarbeitung statt. Die Ausgangssignale einer Schicht bilden die Eingangssignale der Neuronen der nächsten Schicht, bis die Ausgabeschicht erreicht wird. In Neuronen der Ausgabeschicht findet oft nur eine ungewichtete Aggregation statt und der Zustand der Ausgabeschicht $O = (o_{1_s}, \ldots, o_{k_s})$ ist die Ausgabe des MLPs. Grundsätzlich können selbst MLPs mit einer versteckten Schicht dem Universellen Näherungstheorem[17] folgend jede Borel-meßbare Abbildung eines endlich dimensionalen Raum auf einen anderen mit einem beliebig kleinen Fehler approximieren, falls das SLP eine genügend große Anzahl an Neuronen in der versteckten Schicht enthält.

[14] Die oft genutzte `ReLU`-Funktion $ACT_{ReLU}(s) = max(0, s)$ ist zwar an Stelle 0 nicht differenzierbar, hat aber viele Verarbeitungsvorteile und eine höhere Leistung als die vergleichbare `Softplus`-Funktion (vgl. [GBC16]: S. 200). So wird in der Praxis an Stelle 0 entweder die untere oder obere Schranke der Subgradienten genutzt (meist die untere Schranke, vgl. ebd.).

[15] Wobei die Bezeichnung MLP fehlleitet, da die genutzten Neuronen wegen der von der HEAVISIDE-abweichenden Aktivierungsfunktion keine Perzeptronen im engen Sinne mehr sind. Zum anderen sind die Neuronen der Eingabe- und Ausgabeschicht oft nicht-verarbeitend und so für die Berechnung weniger relevant, weshalb MLPs mit nur einer verarbeitenden, versteckten Schicht auch SLP (engl.: *single layer perceptron*) genannt werden.

[16] Wir vereinfachen hier den Aufbau. Schon [RHW86] erwähnt Auslass-Verbindungen (engl.: *skip connections*, wie wir sie später in wiederfinden werden.

[17] Zuerst [HSW89]. Für aktuelle Forschung vgl. [KL20]: S. 1f.

2.2.2 Training und Modelle

Wir fassen in Gewichtsvektoren $w(x) = (w(x)_1, \ldots, w(x)_j)$ die Gewichte der Eingangssignale eines Neurons x einer versteckten Schicht h_i zusammen und die Gewichtsvektoren bilden zusammen die Gewichtsmatrix W_i der Schicht. Die Grundwerte b der Neuronen fassen wir pro Schicht in einem Vektor B_i zusammen.[18] Wir nennen die aktuelle Belegung dieser Parameter das Modell des KNNs. Das Training des Modells ist eine Folge von gezielten Modifizierungen der Parameter mit dem Ziel, dass das KNN nach der Modifizierung unsere gewünschte Abbildung eher approximiert. Dafür müssen wir die Abweichung quantifizieren und aus der Abweichung ableiten können, wie wir die Gewichte und Grundwerte ändern müssen. Die Abweichung wird über eine Verlustfunktion L berechnet. Eine Möglichkeit ist es, die Ausgabe des KNNs für bestimmte Eingaben mit einer von uns vorher gesetzten oder ermittelnden Grundwahrheit (engl.: *ground truth*) für diese Eingabe zu vergleichen und über den Unterschied einen Verlust zu berechnen, z. B. indem wir die Summe der mittleren quadratischen Abweichungen berechnen. Da wir hier die Ausgabe des Netzes mit vorher bekannten Grundwahrheiten vergleichen, bezeichnen wir diese Form des Lernens als überwachtes Lernen (engl.: *supervised learning*). Wir modifizieren das Modell nun indirekt über die Minimierung von L per Gradientenverfahren und Veränderung der Gewichte und Grundwerte über das Fehlerrückführung-Verfahren (engl.: *back-propagation*).[19] Wir setzen dieses aktualisierte Modell wieder unseren Eingaben aus und wiederholen den Prozess. Jede dieser Wiederholungen, in denen wir den Trainingsdatensatz einmal durchlaufen, nennen wir eine Epoche und nach einer bestimmten Anzahl an Epochen oder wenn ein bestimmter Verlustwert unterschritten wird, brechen wir den Trainingsprozess ab und erhalten als Resultat ein trainiertes Modell: eine Parameterbelegung für ein bestimmtes KNN-Design, dass eine bestimmte Abbildung genügend gut approximiert.

2.2.3 Test- und Trainingsdatensätze

Für das Training und um den Trainingserfolg zu messen, benötigen wir beim überwachten Lernen Eingangsinformationen und passende Grundwahrheiten. Wir entwickeln und trainieren ein KNN-Design mit der Absicht, trainierte Modelle für die Verarbeitung spezifischer Zieldaten einzusetzen. Sei es, handschriftlich notierte Zahlen zu erkennen oder Börsensituationen zu verstehen. Leider können wir selten direkt mit dem Zieldatensatz lernen, da er zu groß ist (z. B. alle möglichen handschriftlich notierten Zahlen) und/oder unzugänglich ist (z. B. jede mögliche Situation im Aktien- und Derivatehandel) und/oder wir nur Grundwahrheiten für einen kleinen Teil besitzen (z. B. weil die Berechnung von SE_{id} für große AFs lange dauern kann). Wir trainieren unser Modell deshalb an einem abweichenden Trainingsdatensatz, möchten aber, dass es von diesem unabhängig generalisiert und das Modell

[18] Vgl. im Folgenden [GBC16]: S. 171ff. und 103f.
[19] Mehr dazu in [GBC16]: S. 204ff.

nützlich für den Zieldatensatz ist. So soll das Modell nicht nur die Grundwahrheiten der Trainingseingaben auswendig lernen, sondern Merkmale und Regelmäßigkeiten ableiten. Dafür teilen wir den Trainingsdatensatz, für den wir Eingaben und Grundwahrheiten haben, in den eigentlichen Trainingsdatensatz, einen Abschlusstestdatensatz und eventuell einen Validierungsdatensatz. Der eigentliche Trainingsdatensatz wird zum Training genutzt; wir sagen auch dem Modell sind die enthaltenen Beispiele ‚bekannt'. Der Abschlusstestdatensatz steht für den unzugänglichen Zieldatensatz ein und wir messen an ihm, wie gut unser Modell ungesehene Signale verarbeitet. Ein Validierungsdatensatz hilft, einen geschickten Endpunkt für das Trainieren zu finden[20] oder um ‚Trainingsversagen' aufzuspüren. Dies geht auf Kosten der Größe des eigentlichen Trainingsdatensatz und schwächt damit unter Umständen die Effektivität des Trainings.

Mit steigender Epochenzahl sinkt im Allgemeinen der Verlust, bis der Algorithmus ein Minimum erreicht hat, das er nicht mehr verbessern kann. Von einzelnen Ausreißern abgesehen, hat nun der Verlust ein Plateau erreicht. Prüfen wir nach jeder Epoche mit einem Datensatz mit unbekannte Daten (z. B. einem Validierungsdatensatz), wie gut das angepasste Modell unbekannte Signale verarbeitet, so erhalten wir eine nach unten offene Kurve. Legen wir sie über die Kurve des Trainingsverlustes, so sehen wir, dass sich das Einordnungvermögen von ungesehenen Signalen zunächst noch weiter verbessert, nachdem die Verlustfunktion ihr Plateau erreicht hat, und dann nach einem Leistungsmaximum wieder sinkt, da die Modelle nun unbekannte Signale wieder auf eher ungewünschte Ausgaben abbilden.[21] Dieser letzte Bereich ist der der Überanpassung (engl.: *overfitting*) an den Trainingsdatensatz und zeichnet sich durch die zunehmend schlechtere Generalisierung für unbekannte Signale aus.

Eine Technik, um geschickte Endpunkte auszumachen, heißt „frühes Stoppen" (engl.: *early stopping*)[22] und nutzt einen Validierungsdatensatz (in der Erwartung, dass sich diese Tendenzen auch auf den Abschlusstestdatensatz und später auf den Zieldatensatz übertragen lassen), um Modelle zu finden, die gut generalisieren und nicht überangepasst sind. In seiner einfachsten Form nutzt frühes Stoppen stets das Modell der Epoche, in der das Maximum der Leistungskurve des Validierungstestdatensatz liegt.[23] Weiterhin bezeichnen wir mit ‚Trainingsversagen' Situationen, in denen ein Training in einem oder mehreren Parameterbelegungen konvergiert, die wesentlich schlechtere Leistung als andere zeigen. Wir nennen Trainingsversagen punktuell, wenn die Leistung eines trainierten Modells wesentlich schlechter als die der meisten anderen im gleichen KNN-Design trainierten Modelle ist. Wir nennen es allgemein, wenn die Leistung jedes im gleichen KNN-Design trainierten Modells wesentlich schlechter als die Leistung von in eng verwandten Designs trainierten Modellen ist. Eine Prüfung mit einem Validierungsdatensatz am Ende

[20] Siehe [GBC16]: S. 120.
[21] Vgl. [GBC16]: S. 110ff.
[22] Vgl. [GBC16]: S. 246-252.
[23] Für andere Formen: ebd.

des Trainingsvorgangs spürt viele Fälle von Trainingsversagen auf, bevor der Abschlusstestdatensatz genutzt wird, und ermöglicht es darauf zu reagieren und z. B. das Training bei einem punktuellen Trainingsversagen zu wiederholen, um die Leistung im Abschlusstest zu verbessern. Wir nutzen keinen Validierungsdatensatz, sondern trainieren alle unsere Modelle mit einer festen Epochenzahl von 200, da sich zum einen in einem unserer Datensätze nur sehr wenige AFs bestimmter AF-Typen befinden, so dass die Bildung eines Validierungsdatensatzes unseren Trainingsdatensatz zu sehr verkleinern würde, und zum anderen die Tendenz zu punktuellem Trainingsversagen eine für uns interessante Eigenschaft eines Designs ist, um es von anderen ohne diese Eigenschaft abzugrenzen.

Um Überanpassung entgegenzuwirken nutzen wir zwei andere Techniken: Die Auslass-Technik oder *Dropout* und Trainingswahrscheinlichkeit für Argumente. *Dropout*[24] ist eine Regularisierungstechnik, die der Überanpassung entgegenwirkt, indem für jede Epoche mit einer bestimmten Wahrscheinlichkeit das Ausgabesignal eines künstliche Neurons auf Null gestellt wird. *Dropout* approximiert so rechenzeit- und speicherplatzsparend die Mittelung der Vorhersagen verschiedener KNNs mit unterschiedlicher Struktur, ohne dass diese KNNs getrennt trainiert oder deren Modelle jedes einzeln gespeichert werden müssen. Wir nutzen *Dropout* mit einer Wahrscheinlichkeit von 0,2. Eine andere Möglichkeit, um Überanpassung an den Trainingsdatensatz entgegenzuwirken, ist es, pro Epoche nur die Einordnung eines Teils der Argumente für die Berechnung des Verlustes heranzuziehen.[25] Jede Epoche setzt damit das Modell einem etwas anderen Lerndruck aus, so als wäre es an einem etwas anderen Trainingsdatensatz trainiert worden. Hierfür nutzen wir eine Trainingswahrscheinlichkeit pro Epoche und Argument von 0,8.

2.3 GNNs: Graphen-basierte Neuronale Netzwerke

Graphen-basierte Neurale Netzwerke (kurz GNNs, engl.: *graph neural networks*) sind künstliche neuronale Netzwerke, die die Struktur eines Graphen und die in Knoten und Kanten enthaltenen Informationen nutzen, um neue relevante Informationen über den Graphen, seine Knoten oder Kanten ableiten, bzw. Informationen über das, was diese repräsentieren. Dafür nutzen sie die Kanten zur Nachrichtenübermittlung zwischen Nachbarknoten oder andere Verfahren wie Konvolutionen. GNNs sind dabei eine Verallgemeinerung anderer KNNs wie den in der Bilderkennung genutzten CNNs (engl.: *convolutional neural networks*),[26] in denen Nachbarschaft z. B. nur als ein räumliches Nebeneinander von Pixeln aufgefasst wird. Fassen wir ein Bild als Graphen auf, so ist jeder Pixel ein Knoten mit Kanten zu seinen Nachbarpixeln, wobei z. B. auch in den Kanten die Information gespeichert werden kann, in welcher relativen Position zum mittleren Pixel sich die Nachbarpixel jeweils befinden. In vielen Fällen muss auf solche Weise erst aus einem Objekt der Problemdomäne

[24] Zuerst in [SHK+14]. Zu nachfolgenden Arbeiten vgl. [GBC16]: 258ff.
[25] Wir folgen hier [MYNM20] 50f.
[26] Vgl. [GBC16]: 330ff.

eine Graphstruktur abstrahiert und optional Informationen für Knoten, Kanten und den Graphen extrahiert werden. In unserem Fall sind AFs trivial als Graphen darstellbar und die Knoteninformation, die wir als Eingangsmerkmal jedes Knoten a nutzen, sind die trivial abzuleitenden Mächtigkeiten a^- und a^+. Typische Probleme für GNNs sind neben der für uns relevanten Knotenklassifikation (Ist dieses Argument ia oder ina?) auch Kantenklassifikation (In welcher Beziehung stehen zwei Nutzer einer Soziale-Medien-Plattform zueinander?), Graphenklassifikation (Welche anderen Moleküle haben eine ähnliche Wirkung wie ein bekannter Wirkstoff?), Gruppenbildung (engl.: *node clustering*; zu welchen größeren Interessensgruppen lassen sich Wähler zusammenfassen?) und Verbindungsvorhersage (engl.: *link prediction*; von welchen anderen ihm unbekannten Restaurants wird sich ein Nutzer am ehestens beliefern lassen, wenn seine Lieblingsrestaurants geschlossen sind?). Bei der Knotenklassifikation wird ein Knoten über seinen letzten Zustand klassifiziert. Hier ist die grundlegende Idee, dass jeder Knoten Nachrichten seiner Nachbarn sammelt und über die Verarbeitung der Inhalte dieser Nachrichten seinen eigenen Zustand schrittweise aktualisiert. In der Logik eines MLPs hat jede Schicht dieselbe Anzahl an künstlichen Neuronen wie der Graph Knoten hat. Jeder Nachrichtenübermittlungsschritt entspricht einer neuen Schicht. Der Zustand eines Knotens x nach y Schritten ist $h_x^{y\prime}$. Ein künstliches Neuron verarbeitet nun aber nicht mehr die gewichtete Eingabe aller Neuronen der vorhergehenden Schicht, sondern (da der GNN die Graphstruktur nutzt) nur noch die Ausgabe der Neuronen, die Knoten repräsentieren, von denen eine Kante zu diesem Knoten ausgeht. Im Vergleich zu einem klassischen MLP mit seiner dichten Vernetzung ist die des typisches GNNs mager (engl.: *sparse*). Für die ersten Versuche maschinelles Lernen mit abstrakten Argumentationssystemen zu verbinden, nutzten KUHLMANN und THIMM `FM2`. `FM2` ist ein flaches GNN, das das Ergebnis nach zwei Konvolutionschichten oder GCN-Schichten (engl.: *graph convolutional network*) nach KIPF und WELLING zur Klassifikation von Argumenten nutzt.[27] Eine KIPF-WELLING-Konvolution approximiert einen Glättungsfilter in der spektralen Domäne über eine vereinfachte K-verortete Konvolution nach HAMMOND et al. mit K = 1 und einem Renormalisierungstrick.

In spektralen GNN-Verfahren wird die Konvolution in der spektralen Domäne definiert:

$$g \star x = \mathcal{F}^{-1}(\mathcal{F}(g) \odot \mathcal{F}(x)).$$

Dazu wird zunächst das Signal des Graphen x aus $\mathbb{R}^N$ per Graph-Fourier-Transformation $\mathcal{F}$ in die spektrale Domäne transformiert:

$$\mathcal{F}(x) = U^T x \text{ und } \mathcal{F}^{-1}(x) = Ux.$$

U ist Ergebnis der Zerlegung der normalisierten LaPlace-Kirchhoff-Matrix L in die Matrix der Eigenvektoren U und die Diagonalmatrix der Eigenwerte Λ.

$$L = U\Lambda U^T.$$

[27] Vgl. im Folgenden [KW17] und [ZCH+20].

Dies ist möglich, da L symmetrisch und positiv semidefinit ist. L wird aus der Einheitsmatrix I, der Gradmatrix D und der Adjazenzmatrix A berechnet:

$$L = I_N - D^{-0.5}AD^{-0.5}.$$

Damit ist die Konvolution (mit Mallat [Mal08]) definiert als

$$g \star x = \mathcal{F}^{-1}(\mathcal{F}(g) \odot \mathcal{F}(x)) = U(U^T g \odot U^T x)$$

mit $U^T g$ als Filter. Dieser wird zu $g_\theta = diag(\theta)$ vereinfacht, mit θ aus $\mathbb{R}^N$ als lernbarem Parameter.

$$g_\theta \odot Lx = U g_\theta U^T x.$$

HAMMOND et al. [HVG11] approximieren $g_\theta(\Lambda)$ über eine abgebrochene Tschebyscheff-Reihe.

$$g_{\theta'}(\Lambda) \approx \Sigma_{k=0}^{K} \theta'_k T_k(\tilde{\Lambda}).$$

$\tilde{\Lambda}$ ist dabei $\frac{2}{\lambda_{max}}\Lambda - I_N$ mit λ_{max} dem höchsten Eigenwert von L. Damit ist θ' aus $\mathbb{R}^K$ der Vektor der Tschebyscheff-Koeffizienten. In die Konvolution eingesetzt erhalten wir

$$g_{\theta'} \star x \approx \Sigma_{k=0}^{K} \theta'_k T_k(\tilde{L}).$$

$\tilde{L}$ ist dabei $\frac{2}{\lambda_{max}}L - I_N$. Dieser Ausdruck ist K-lokalisiert, d.h. der Wert eines Knotens ist nur von Knoten abhängig, die nicht weiter als K Kanten entfernt liegen. KIPF und WELLING begrenzen nun K auf genau 1 und approximieren λ_{max} auf 2. Damit vereinfacht sich die Konvolution zu

$$g_{\theta'} \star x \approx \theta'_0 x + \theta'_1 (L - I_N)x = \theta'_0 x + \theta'_1 D^{-0.5}AD^{-0.5}x.$$

Um Überanpassung zu vermeiden und als Rechenzeiteinsparung wird $\theta'_0 = -\theta_1 = \theta$ gesetzt:

$$g_{\theta'} \star x \approx \theta(I_N + D^{-0.5}AD^{-0.5})x.$$

Durch einen Renormalisierungstrick wird die Operation gutmütiger. Statt $I_N + D^{-0.5}AD^{-0.5}$ nutzen KIPF und WELLING $\tilde{D}^{-0.5}\tilde{A}\tilde{D}^{-0.5}$ mit $\tilde{A} = A + I_N$ und $\tilde{D}_{ii} = \Sigma_j \tilde{D}_{ij}$.

Führen wir diese Operation mehrfach hintereinander aus, jeweils verbunden mit einer nicht-linearen Aktivierungsfunktion σ, erhalten wir als Propagationsregel einer GCN-Schicht nach KIPF und WELLING:

$$H^{(l+1)} = \sigma(\tilde{D}^{-0.5}\tilde{A}\tilde{D}^{-0.5}H^{(l)}W^{(l)})$$

$W^{(l)}$ ist eine schichtspezifische trainierbare Gewichtsmatrix und $H^{(l)}$ ist die Matrix der Aktivierungen in der l-ten Schicht. Die Eingabe-Schicht ist definiert als die Matrix der Eingabemerkmale: $H^{(0)} = X$.

Knoten eines GCNs mit k Schichten erhalten Informationen aus der k-Hop-Nachbarschaft. Damit agglomerieren sie u. U. relevante Informationen, zu denen sie ansonsten keinen Zugang hätten.[28] Allerdings leiden tiefe GCNs an Überglättung (engl.: *oversmoothing*).[29] Die Intuition ist, dass, nachdem ein Glättungsfilter oft genug angewendet wurde, sich alle (verbundenen) Knoten aneinander angeähnelt haben und dabei u. U. für uns relevante Informationen verlieren, obwohl sie Informationen aus größerer Distanz miteinbeziehen. Per se ist ein Konvergieren in einen Unterraum für die Knotenklassifikation von AFs kein Problem, sondern gewollt, solange die Werte im Unterraum über die Ausgabeschicht dann eine möglichst korrekten Zuordnung zu ia und ina (bzw. pa und pna, bzw. pa* und pna*) ermöglichen. Dies ist aber nicht garantiert. OONO und TAIJI haben nun zumindest für genügend große und dichte Erdős-Rény-Graphen gezeigt, dass deren Knotenmerkmale bei einem unendlich tiefen GCN tatsächlich dazu tendieren, in einem Unterraum zu konvergieren.[30] Knoten mit demselben Knotengrad in einer Zusammenhangskomponente konvergieren in demselben Wert. Nur die Information über Zugehörigkeit zu einer Zusammenhangskomponente und sowie der Knotengrad bleiben erhalten. Da die meisten AFs aber relativ wenig Kanten besitzen (und nur ein Teil Erdős-Rény-Graphen sind), ist dieser Befund nicht direkt übertragbar. Komplexere GCNs nutzen gegen Überglättung Maßnahmen wie Rohresiduen (engl.: *initial residuals*) oder Identitätsabbildung (engl.: *identity mapping*).[31] Aber auch bei GCNs mit Residuen zeigt sich die Tendenz zur Konvergenz mit Informationsverlust: „Such convergence leads to over-smoothing as the vector [of convergence] π only carries the two kinds of information: the degree of each node, and the inner product between the initial signal x and vector $D^{0,5}\vec{1}$."[32] Insgesamt macht dies die Entwicklung komplexer und die Berechnung langsamer. Wir nutzen keine dieser Techniken zur Ermöglichung von tieferen GCNs.

Ein anderes Problem von GCNs ist, dass sie sich am besten für homophile Datensätze eignen, bei denen Kanten eine gewisse Form von Nähe oder Ähnlichkeit

[28] In unserem Falle ist es zum Beispiel für die Zugehörigkeit eines Argumentes zu idealen Extension relevant, ob ein Nachbarargument schon sicher in dieser Extension ist oder eben nicht.

[29] LI et al. gehen noch weiter und beschreiben die GCN-Konvolution als Spezialfall einer Laplace-Glättung: „In particular, we show that the graph convolution of the GCN model is simply a special form of Laplacian smoothing, which mixes the features of a vertex and its nearby neighbors. The smoothing operation makes the features of vertices in the same cluster similar, thus greatly easing the classification task, which is the key reason why GCNs work so well. However, it also brings potential concerns of over-smoothing. If a GCN is deep with many convolutional layers, the output features may be oversmoothed and vertices from different clusters may become indistinguishable. The mixing happens quickly on small datasets with only a few convolutional layers [...]" ([LHW18] S. 3538f.).

[30] „In the context of node classification tasks, we can interpret this corollary as the ‚information loss' of GCNs in the limit of infinite layers. For any $X \in M$, if two nodes $i, j \in V$ are in a same connected component and their degrees are identical, then, the column vectors of X that correspond to nodes i and j are identical. It means that we cannot distinguish these nodes using X"([OS19]: S.5.).

[31] Vgl. zu den Techniken [CWH$^+$20]: S. 1727f.

[32] [CWH$^+$20]: S. 1729. Dies verläuft schneller mit Knoten mit hohem Knotengrad (ebd. S. 1730).

bedeuten.[33] Und diese Homophilie ist für AFs nicht gegeben. Für unser Klassifikationsproblem benötigen wir am Ende eine Aussage darüber, inwiefern ein Argument eher ia oder ina, pa oder pna oder pa* oder pna* ist. Bei diesen Eigenschaften ist der Einfluss der direkten Nachbarschaft differenzierter. Eine Kante allein kann zwischen zwei Argumenten je nach Eigenschaft Ähnlichkeit, Differenz oder einen uneindeutigen Zwischenzustand bedeuten, der weitere Informationen zur Klärung benötigt. Zuallerest bedeutet eine verbindende Kante zwischen Argument a nach b einen Angriff, weshalb es ausgeschlossen ist, dass a mit einem anderen Argument $b \in a^+ \cup a^-$ zusammen in $S = E_{id}$ sein können. S wäre ansonsten nicht konfliktfrei und folglich $S \notin \text{adm}(AF)$. Dasselbe gilt für pa. Angreifer und Angegriffener können nicht beide gleichzeitig ia bzw. pa sein. Nachbarschaft bedeutet für diese beiden Eigenschaften Differenz. Allerdings könnten beide Argumente pa* sein, falls sie sich jeweils exklusiv in einer von zwei nicht deckungsgleichen bevorzugten Extensionen befinden. Falls beide pa* sind, ist keines pa. Die Wahrscheinlichkeit, dass zwei zufällig ausgewählte Argumente beide pa* sind, ist demensprechend geringer, falls es zwischen ihnen einen Angriff gibt, da dann alle Fälle ausgeschlossen sind, in denen ein Argument oder beide pa* und pa sind. Für diese Eigenschaft bedeutet direkte Nachbarschaft eine Tendenz zur Differenz ohne abschließendes Urteil. Trivialerweise hingegen können Angreifer und Angegriffener beide ina, pna oder pna* sein, müssen es aber nicht. Über die Angriffsrelation wird die Eigenschaft nicht weitergeleitet.[34] Damit bedeutet eine Kante zwischen zwei Knoten nicht per se Ähnlichkeit, sondern auch das Gegenteil, obwohl sich über die GCN-Konvolution ihre Merkmalswerte annähern.

Erweitern wir die Nachbarschaft auf die 2-Hop-Nachbarschaft eines Knoten, wird ein weiteres Problem der GCN-Konvolution deutlich. Nehmen wir als Beispiel drei Argumente a, b und c mit $(a, b), (b, c) \in R$. Die Argumente a und c sind in einer Verteidiger-Verteidigter-Relation. Beide können folglich in jeder Konfiguration ia und ina, bzw. pa und pna oder pa* und pna* sein. Relevant wird diese Eigenschaft deshalb, da die Verteidigung eine Grundeigenschaft auch jeder Extension E_{id} und E_{pr} ist. Das Mittelargument b, das Teil von a^+ und Teil von c^- ist, kann in diesem Falle als direkte Nachbarn von b nie zusammen mit a oder c ia bzw. pa sein. Nur die Werte der beiden Randargumente sollten sich nun aneinander anähneln. Dies geschieht aber in unserem Beispiel nur über b: über die Anähnelung der Merkmalswerte des Mittelargumentes. Die Merkmalswerte von b sollen sich aber von den Randargumenten eher unterscheiden. Noch schwieriger wird es dadurch, dass das gerichtete AF über die $\tilde{D}^{-0.5}\tilde{A}\tilde{D}^{-0.5}$-Normalisierung eines GCN zu einem ungerichteten Graphen mit Schlingen an jedem Knoten wird. Jede Information darüber, wer wen angreift, geht verloren. a^+ und a^- werden ununterscheidbar vereint. Damit geht die Information über die Verteidiger-Verteidigter-Relation in einer all-

[33] Viele GNN-Architekturen wie GCN „implicitly assume homophily in the graph, where most connections happen among nodes in the same class or with alike features“([ZRR+21]: 11168).

[34] Allerdings macht eine hohe Anzahl an ein- und ausgehenden Angriffen es in AF-Datensätzen, die den von uns verwendeten gleichen, wahrscheinlicher, dass ein Argument pna ist ([KWT22]: S. 232.).

gemeinen, ungerichteten 2-Hop-Nachbarschaft unter. KIPF selbst schlägt in einem Github-Kommentar als alternative Normalisierung für Graphen, in denen die Richtung der Kanten wichtig ist (im konkreten Fall ging es um Wissensgraphen), eine Normalisierung mit $D^{-1}A$ vor.[35] Wir werden aber eine andere alternative Schicht mit den klassischen GCN-Schichten vergleichen.

Weiterhin geht durch die neuen Schlingen, die Information über sich selbst angreifende Argumente verloren, die sich wegen nicht gegebener Konfliktfreiheit aus ia(AF) und pa(AF) ausschließen.[36] Andere GNNs wie das Nachrichtenübermittlungs-GNN AGGN[37] haben letztere Probleme wegen der fehlenden Normalisierung nicht. Das AGNN nutzt sogar ein minimal heterogenes GNN, indem Nachrichtenübermittlung sowohl in als auch entgegen der Richtung der Angriffsrelation als getrennte Operationen implementiert wurden.[38] Eine Alternative zu einem klassischen GCN wäre es, ein AF als heterogenen Graphen mit zwei gerichteten Kantentypen für Angriff und Verteidigung zu beschreiben. Eine solche Konvolution muss von der von KIPF und WELLING abweichen, wie z. B. die R-GCN-Konvolution ([SKB+17]).

Wegen dieser Probleme vergleichen wir die Leistung von GNNs mit KIPF-WELLING-normierten GCN-Schichten mit der von GCN-Schichten ohne diese Normalisierung, so dass Gerichtetheit und Schlingen eines AFs erhalten bleiben. Im Vergleich zu einem KIPF-WELLING-normierten GCN oder einem AGNN fließen Informationen aber nur in Angriffsrichtung, was die Ausbreitung von Informationen im Allgemeinen verlangsamt. Die Propagationsregel unserer allgemeineren Schicht ist:

$$H^{(l+1)} = \sigma(AH^{(l)}W^{(l)}).$$

Für unsere Experimente vergleichen wir Designs dieser beiden Architekturen mit zwei bis zehn GNN-Schichten. Die übliche Auslesefunktion besteht dabei aus `MEAN` über den Vektor h_x^{out} für jedes Argument mit anschließenden `SIGMOID`. Diesen Wert zwischen 0 und 1 runden wir und erhalten eine Klassifikation, ob ein Argument ia oder ina ist. Wir bemerkten dabei in Vortests Schwächen in der Interpretation der Endzustände von GCNs bei AFs, die AGNNs nicht aufwiesen. Durch die MLP-Auslesefunktion des AGNNs motiviert, nutzen wir anstatt der üblichen Auslesefunktion zusätzlich Designs mit ein bis fünf lineare Schichten, wobei die letzte Schicht stets eine Ausgabedimension von 1 hat.[39] Abschließend folgt erneut `SIGMOID`.

Wir benennen die Designs nach der Art (GCN oder GNN) und Tiefe (2-10) der GNN-Schicht und Tiefe (0-5) der linearen Schicht. Ein Design mit drei KIPF-WELLING-GCN-Schichten und 5 linearen Schichten nennen wir `GCN3LIN5` und ein anderes

[35] https://github.com/tkipf/gcn/issues/91/

[36] Besonders relevant für SemBuster-Graphen (siehe Abschnitt 2.5.), die stets $|A|/3$ Argumente mit Schlingen haben, deren Information durch die GCN-Konvolution verloren geht.

[37] Siehe Abschnitt 3.3.

[38] Die Erhaltung der Gerichtetheit und ursprünglichen Schlingen können zwei Gründe für die gute Leistung von AGNNs gegenüber GCNs sein.

[39] Wir stellen die Frage hintan, ob bei einem GNN mit zwei GCN-Schichten und fünf lineare Schichten die linearen Schichten noch eine besondere Auslesefunktion sein können, oder ob dieses GNN nicht eher ein KNN mit linearen Schichten und einer zwei-schichtigen GCN-Vorverarbeitungsfunktion ist.

Design mit acht Schichten ohne die Normalisierung und ohne lineare Schichten nennen wir schlicht `GNN8LIN0`. Wir vergleichen damit in jedem unserer Experimente 108 Designs mit insgesamt 2 bis 15 KNN-Schichten miteinander. Dadurch können wir den Einfluss der fehlenden Normalisierung, den der Anzahl von GNN-Schichten und den der unterschiedlichen, sowie unterschiedlich tiefen Auslesefunktionen auf die Leistung untersuchen.

2.4 Leistungskennnzahlen: MCC, TPR, TNR, ACC

Um die Leistung eines KNNs einzuschätzen, nutzen wir verschiedene Kennzahlen, die aus der Wahrheitsmatrix abgeleitet werden. Wir nennen Argumente, die ia (pa, pa*) sind, die positiven Fälle und Argumente, die ina (pna, pna*), die negativen Fällen. Die Anzahl der korrekt als positiv eingestuften positiven Fälle nennen wir *TP* (engl. *true positive*), die Anzahl der inkorrekt als negativ eingestuften positiven Fälle *FN* (engl. *false negative*), die Anzahl der inkorrekt als positiv eingestuften negativen Fälle heißt *FP* (engl. *false positive*) und die Anzahl der korrekt als negativ eingestuften negativen Fälle *TN* (engl. *true negative*). Diese vier Anzahlen werden in einer Wahrheitsmatrix zusammengefasst und die Kennzahlen aus ihnen abgeleitet. Die Gesamtgenauigkeit oder *ACC* (engl. *accuracy*) gibt den Anteil der korrekt eingestuften Fälle (die Summe aus TP und TN) an allen Fällen an. Bei einer unausgeglichenen Klassenverteilung ist ACC als Hauptkennzahl weniger aussagekräftig. So zeigen Nein-Entscheider[40] in einem unserer Datensätze eine Genauigkeit von 0,8 bis 0,9. Die Sensitivität oder *TPR* (engl. *true positive rate*) gibt den Anteil der korrekt eingestuften positiven Fälle TP an allen positiven Fällen *P* (engl. *positives*: die Summe aus TP und FN) an. Die Spezifität oder *TNR* (engl. *true negative rate*) gibt den Anteil der korrekt eingestuften negativen Fälle TN an allen negativen Fällen *N* (engl. *negatives*: die Summe aus TN und FP) an. Die Spezifität ist in den untersuchten Modellen im Allgemeinen[41] höher als die Sensitivität.[42] Die Sensitivität variiert oft stark und leistungsstarke Modelle zeigen im Allgemeinen eine relativ hohe Sensitivität und sehr hohe Spezifität.

Der MCC (engl. *Matthews correlation coefficient*[43]) ist besonders für unausgeglichene Datensätze geeignet, um in einem Wert die Leistung eines Modells einzufangen. Dies ist für uns wichtig, da wir die vielen Ergebnisse unserer Experimente für Leser verständlich aufarbeiten müssen. Die Fokussierung auf einen MCC-Wert pro Design statt mehrerer Kennzahlen ermöglicht es uns, die Leistungsunterschiede vieler GNN-Designs nicht nur in einer mehrseitigen Tabelle, sondern auch komprimiert in

[40] Ja- und Nein-Entscheider sind Extremfälle unter den möglichen Modellen, die jedes Argument stets entweder als ia (pa, pa*) bzw. stets als ina (pna, pna*) einordnen. Im Allgemeinen deutet die Degeneration eines Modells zu einem Entscheider auf ein Trainingsversagen hin.

[41] Ein Gegenbeispiel sind viele Modelle der KIPF-WELLING-GCN-Designs in Experiment $E(kwt)$ (siehe Abschnitt 5.6).

[42] Der Hauptgrund dafür ist, dass unsere Datensätze eine unausgeglichene Klassenverteilung haben und P < N ist.

[43] Der MCC ist identisch mit dem φ-Koeffizienten Pearsons für binäre Klassifikation.

kleineren farbigen Abbildungen darzustellen. Der MCC nimmt Werte zwischen -1,0 und 1,0 an, wobei 1,0 für eine perfekte Vorhersage steht, -1,0 für eine stets inkorrekte Vorhersage und 0,0 für eine von zufälliger Wahl nicht unterscheidbare Vorhersage. Er ist definiert als:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Damit ist der MCC undefiniert, falls mindestens eine der Summen unter der Wurzel $(TP + FP)$, $(TP + FN)$, $(TN + FP)$ oder $(TN + FN)$ gleich 0 ist. Wir notieren dann als Zeichen für einen nicht validen Wert ein ‚–'-Zeichen. Für verschiedene Fälle der nicht validen Werte fügen wir ergänzende Informationen in Klammern hinzu. Betrachten wir den Fall, dass $TP = FP = FN = 0$ und $TN \neq 0$ sei (alle Argumente des AFs sind ina (pna, pna*) und wurden korrekt eingeordnet).[44] Der MCC ist für diese perfekte Vorhersage leider undefiniert. Die nächstbeste, ebenfalls perfekte Vorhersage mit einem gültigen MCC-Wert hätte genau ein korrekt eingeordnetes positives Argument und einen MCC von 1,0. Diesen nächstbesten Wert notieren wir in Klammern. Wir notieren ebenfalls 1,0 für den Fall, dass $TP \neq 0$ und $TN = FP = FN = 0$ sind. Für die Fälle, dass nur FP bzw. $FN \neq 0$ sind, also dass alle Argumente entweder ia bzw. ina sind und keines korrekt eingeordnet wurde, notieren wir -1,0 in Klammern. Falls ein Modell zu einem Ja- bzw. Nein-Entscheider degeneriert ist und die Argumente nicht nur einer Klasse angehören, ist der MCC wegen $TN+FN = 0$ bzw. $TP+FP = 0$ undefiniert und wir notieren ‚ja' bzw. ‚nein' in Klammern hinter dem ‚–'-Zeichen. Im Vergleich zu ACC, TPR und TNR bleiben im MCC andere Informationen der Wahrheitsmatrix erhalten und können, da es sich um einen Wert handelt, einfacher verglichen werden. Falls kein MCC-Wert angegeben wird, kann er über TPR, TNR und ACC näherungsweise[45] berechnet werden. Über $ACC = TPR \times \frac{P}{P+N} + TNR \times \frac{N}{P+N}$ finden wir das Verhältnis zwischen P und N heraus (und damit können wir auch den Anteil an ia, pa bzw. pa* Argumenten des Datensatzes fundiert abschätzen). TPR ist das Verhältnis zwischen TP und FN, und TNR das zwischen TN und FP. Setzen wir einen der Werte der Wahrheitsmatrix fest (z. B. TP auf 1), folgen die anderen aus den Verhältnissen (je nach Wahl des ersten Wertes sind die anderen Werte dann allerdings keine ganzen Zahlen mehr, was aber das Ergebnis des MCCs nicht beeinflusst). Damit können wir dann den MCC berechnen. Wir nutzen diese Methode in Fällen, in denen Autor*innen keinen MCC-Wert selbst berechnet haben, um eine vergleichbare Ergebnisgrundlage zu schaffen.

Allerdings verhält sich der MCC nicht immer wie erwartet. Betrachten wir drei Beispiele in den Tabellen 1-3. Unser Datensatz besteht hier stets aus 10.000 Argumenten und in jedem der Beispiele variieren wir andere Parameter. Wir runden dabei die Kennzahlen ausnahmsweise auf vier Nachkommastellen, um auch kleine

[44] Dies ist z. B. bei der Betrachtung von Ergebnissen für die AFs des SemBuster-Typs der Fall (siehe Abschnitt 2.5).

[45] Dadurch, dass die Kennzahlen zu Publikationszwecken gerundet werden müssen, bleibt der tatsächliche MCC außerhalb unseres Zugriffs.

Schwankungen sichtbar zu machen.

TP	FN	TN	FP	MCC	TPR	TNR	ACC	Anmerkungen
1	1	9998	0	**0,7071**	0,5000	1,0000	0,9999	
1	1	9997	1	**0,4999**	0,5000	0,9999	0,9998	
1	1	9995	3	0,3534	0,5000	0,9997	0,9996	
1	1	9988	10	**0,2129**	0,5000	0,9990	0,9989	
1	1	9965	33	0,1207	0,5000	0,9967	0,9966	
1	1	9898	100	**0,0693**	0,5000	0,9900	0,9899	
1	1	9665	333	0,0367	0,5000	0,9667	0,9666	
1	1	8998	1000	0,0188	0,5000	0,9000	0,8999	
1	1	4999	4999	0,0000	0,5000	0,5000	0,5000	Zufallsentscheider
0	2	9998	0	– (nein)	–	1,0000	0,9998	Nein-Entscheider

Tabelle 1: Im ersten Beispiel halten wir die TP und FN auf 1 und erhöhen langsam FP. Zum Vergleich sind ein idealer Zufallsentscheider und Nein-Entscheider mitaufgeführt.

Im ersten Beispiel in Tabelle 1 setzen wir TP und FN auf 1 und erhöhen FP schrittweise (und senken dadurch TN). Wir sehen, dass für FP = 0 der MCC bei 0,7071 liegt, obwohl nur eines von 10.000 Argumenten falsch eingeordnet wurde. Allerdings bildet dieses eine Argument die Hälfte der positiven Klasse. Bei FP = 1 sinkt der MCC um 0,2 auf knapp unter 0,5, obwohl zusätzlich nur eines von 9998 Argumenten der negativen Klasse nun falsch eingeordnet wird. Bei FP = 10 beträgt der MCC nur noch 0,2129. Bei FP = 100 liegt er schon unter 0,1, obwohl nur 1% der Argumente der negativen Klasse falsch eingeordnet wird. Bei so niedrigen MCC-Werten wird eine Unterscheidung dieser Modelle von einem Zufallsentscheider schwieriger, obwohl von FP = 0 bis FP = 100 sich die TNR nur von 1,0 auf 0,99 verschlechtert hat, während die TPR stabil bei 0,5 lag.

Im zweiten Beispiel in Tabelle 2 setzen wir FN fest auf 1 und FP auf 100 und erhöhen TP schrittweise (und senken dadurch TN). Die ACC bleibt dabei stets stabil bei sehr hohen 0,9899. Wir starten bei TP = 1 und einem MCC von 0,0693. Bei TP = 10 ist der MCC immer noch niedrig bei 0,2857, obwohl die Erkennungsleistung des Modells der positiven wie negativen Klasse mit einer TPR von 0,9091 und einer TNR von 0,99 gut ist. Bei TP = 100 sind TPR und TNR beide knapp bei 0,99, die Erkennungsleistung des Modells ist also für unser Verständnis sehr gut, der MCC liegt aber nur bei moderaten 0,6999. Erst bei TP = 200 sehen wir einen MCC-Wert von über 0,8. Bei TP = 1000 sind knapp 10% der Argumente in der positiven Klasse und die MCC-Werte stabilisieren sich auf einem Niveau über 0,94.

Im dritten Beispiel in Tabelle 3 (S. 19) halten wir die TPR und TNR stabil, erhöhen in großen Stufen die Zahl der Argumente in der positiven Klasse P und erhöhen in kleinen Stufen FP. Wir sehen zum einen, dass je kleiner P ist, desto heftiger reagiert der MCC bei Änderungen der TNR. Zum anderen haben wir zwei MCC-Paare mit a) und b) markiert, die bei ähnlichem MCC, ähnlicher TPR und ähnlicher TNR eine ganz unterschiedliche Erkennungsleistung zeigen, wie der Unterschied in ACC

TP	FN	TN	FP	MCC	TPR	TNR	ACC	Anmerkungen
1	1	9898	100	**0,0693**	0,5000	0,9900	0,9899	
2	1	9897	100	0,1132	0,6667	0,9900	0,9899	
3	1	9896	100	0,1466	0,7500	0,9900	0,9899	
4	1	9895	100	0,1741	0,8000	0,9900	0,9899	
5	1	9894	100	0,1978	0,8333	0,9900	0,9899	
7	1	9892	100	0,2377	0,8750	0,9900	0,9899	
10	1	9889	100	**0,2857**	0,9091	0,9900	0,9899	
20	1	9879	100	0,3962	0,9524	0,9900	0,9899	
35	1	9864	100	0,4994	0,9722	0,9900	0,9899	
60	1	9839	100	0,6041	0,9836	0,9899	0,9899	
100	1	9799	100	**0,6999**	0,9901	0,9899	0,9899	
200	1	9699	100	**0,8102**	0,9950	0,9898	0,9899	
350	1	9549	100	0,8760	0,9972	0,9896	0,9899	
600	1	9299	100	0,9200	0,9983	0,9894	0,9899	
1000	1	8899	100	**0,9476**	0,9990	0,9889	0,9899	
2000	1	7899	100	0,9695	0,9995	0,9875	0,9899	
3500	1	6399	100	0,9782	0,9997	0,9846	0,9899	
18	18	4982	4982	0,0000	0,5000	0,5000	0,5000	Zufallsentscheider
0	36	9964	0	– (nein)	–	1,0000	0,9964	Nein-Entscheider

Tabelle 2: Im zweiten Beispiel halten wir die FN auf 1 und FP auf 100 und erhöhen langsam TP.

zeigt. Das Paar a) zeigt ACC-Werte von 0,9989 und 0,8999, was im Kontext auf eine sehr gute Erkennungsleistung des ersten Modells und eine nur gute des zweiten deutet. Das Paar b) zeigt eine höhere Differenz mit ACC-Werten von 0,998 und 0,6664, was im Kontext auf eine sehr gute Erkennungsleistung des ersten Modells und eine moderate des zweiten deutet.

Wir halten fest: Wenn wir MCC-Werte beim Testen gerade von unterschiedlichen Datensätzen vergleichen, können sich hinter demselben MCC-Wert Modelle mit ganz unterschiedlicher Erkennungsleistung verbergen oder sich hinter verschiedenen MCC-Werten Modelle mit vergleichbarer Erkennungsleistung verbergen. Dies geschieht eher, je unausgeglichener die Klassenverteilung innerhalb und zwischen den Datensätzen ist und je kleiner einzelne der Werte der Wahrheitsmatrix werden. Nur indem wir ACC und u. U. TPR und TNR heranziehen, können wir überraschende oder fragwürdige MCC-Ergebnisse einordnen.

2.5 Die Typen abstrakter Argumentationssysteme

Die AFs, die in der bisherigen Forschung aber auch von uns genutzt werden, lassen sich in zwölf Typen aufteilen. Drei dieser Typen werden über Generatoren des Benchmarkingpakets `probo`[46] generiert: `GroundedGenerator` generiert AFs mit

[46] Siehe [COS+14] und https://sourceforge.net/projects/probo/

TP	FN	TN	FP	MCC	TPR	TNR	ACC	Anmerkungen
1	1	9997	1	0,4999	0,5000	0,9999	0,9998	
1	1	9988	10	0,2129	0,5000	0,9990	0,9989	
1	1	9898	100	0,0693	0,5000	0,9900	0,9899	
10	10	9979	1	**0,6738**	0,5000	0,9999	**0,9989**	a)
10	10	9970	10	**0,4990**	0,5000	0,9990	**0,9980**	b)
10	10	9880	100	0,2099	0,5000	0,9900	0,9890	
50	50	9899	1	0,6983	0,5000	0,9999	0,9949	
50	50	9890	10	0,6429	0,5000	0,9990	0,9940	
50	50	9801	99	0,4024	0,5000	0,9900	0,9851	
100	100	9799	1	0,6999	0,5000	0,9999	0,9899	
100	100	9790	10	0,6698	0,5000	0,9990	0,9890	
100	100	9702	98	0,4924	0,5000	0,9900	0,9802	
500	500	8999	1	0,6874	0,5000	0,9999	0,9499	
500	500	8991	9	0,6811	0,5000	0,9990	0,9491	
500	500	8910	90	0,6239	0,5000	0,9900	0,9410	
1000	1000	7999	1	**0,6662**	0,5000	0,9999	**0,8999**	a)
1000	1000	7992	8	0,6630	0,5000	0,9990	0,8992	
1000	1000	7920	80	0,6315	0,5000	0,9900	0,8920	
3333	3333	3333	1	0,4997	0,5000	0,9997	0,6666	
3333	3333	3331	3	**0,4990**	0,5000	0,9991	**0,6664**	b)
3333	3333	3301	33	0,4889	0,5000	0,9901	0,6634	

Tabelle 3: Im dritten Beispiel halten wir die TPR und TNR stabil und erhöhen P in den großen Stufen und FP in kleinen Stufen. In den mit a) sowie b) markierten Fälle gleichen sich TPR, TNR sowie MCC, aber die ACC weicht stark ab.

einer großen grundierten Extension; `SccGenerator` generiert AFs mit vielen starken Zusammenhangskomponenten; und `Stable- Generator` generiert AFs mit vielen stabilen Extensionen. Drei weitere Typen werden mit dem Paket `AFBenchGen`[47] generiert und nutzen klassische Graphenmodelle, um AFs mit abweichenden Verbindungsstrukturen zu bilden. AFs nach dem *Barabási-Albert*-Modell sind skalenfreie Netze, d.h. die Anzahl der Knotengrade ist besonders ungleich mit einigen wenigen besonders gut vernetzten Knoten. Graphen nach dem *Erdős-Renyi*-Modell zeigen dagegen zufällige, Poisson-verteilte Knotengrade und eine geringe lokale Gruppenbildung. Die Knoten in Graphen generiert nach dem *Watts-Strogatz*-Modell haben dagegen einen hohen Clusterkoeffizienten. Wir kürzen dieser sechs AF-Typen wie [KWT22] mit dem Kürzel *pbbg* ab.

Die AFs aus dem Datensatz des ICCMA-2017-Wettbewerb werden auch von vielen Autor*innen genutzt.[48] Der Datensatz enthält neben den bisher vorgestellten Typen fünf weitere, von denen zwei aus Realweltproblemen abgeleitet worden sind, zwei andere bestimmte Implementationseigenschaften von AF-Lösern prüfen sollen und der letzte aus ABA-Systemen abgeleitet wurde.[49] Abgeleitete Typen sind `Planning2AF` und `Traffic`. Die AFs in `Planning2AF` sind Planungsprobleme, die auf der Bewegung eines Roboterarms und Interaktion mit Blöcken sowie auf dem Betrieb einer Autofähre zwischen Inseln fußen(vgl. [CGV]). Trotz des begrenzten Rahmens sind die abgeleiteten AFs sehr unterschiedlich: $|A|$ rangiert zwischen 86 und knapp 2000 und $|R|$ zwischen 136 und 4255. Für `Traffic`-AFs wurden realexistierende Verkehrsnetze zu AFs transformiert, indem jede (ungerichtete) Kante entweder mit einer pro AF festen Wahrscheinlichkeit von 20%, 50% oder 80% durch zwei gegenseitige Angriffe ersetzt wird oder durch einen Angriff mit zufälliger Richtung ersetzt wird(vgl. [Dil]). Die restlichen drei ICCMA-Typen sind `ABA2AF`, `AdmBuster` und `SemBuster`. `ABA2AF`-AFs sind aus generierten ABA-Systemen AFs ab und haben eine hohe Angriffsdichte (vgl. [LWJ]). `AdmBuster`-AFs haben genau eine grundierte Extension, die deckungsgleich mit der idealen und der einzigen bevorzugten Extension ist (vgl. [CP]). Diese Extension enthält die Hälfte aller Argumente. `SemBuster`-AFs haben keine ideale Extension und $|A|/3+1$ bevorzugte, wobei kein Argument in jeder Extension vorkommt. Ein Drittel aller Argumente greift sich selbst an, der andere zwei Drittel sind pa* (vgl. [CV]). Eine Menge von AFs, die im ICCMA-2017-Wettbewerb genutzt wurde, bezeichnen wir mit dem Kürzel *iccma*.

Der jüngste Zugang sind AFs des `KWT-Generators` aus [KWT22], der AFs mit einer im Vergleich zu seiner idealen Extension kleinen grundierten Extension erzeugt, um die Lösungsschwierigkeit zu erhöhen.[50] Für Mengen mit kwt-AFs nutzen wir das Kürzel *kwt*.

[47] Siehe [GCV14] und https://sourceforge.net/projects/afbenchgen/

[48] ICCMA steht für International Competition on Computational Models of Argumentation und ist unter http://argumentationcompetition.org zu finden.

[49] Zur Zusammenstellung der AFs im ICCMA-2017-Wettbewerb, vgl. [GLMW].

[50] Der Generator kann unter http://tweetyproject.org/r/?r=kwt_gen gefunden werden.

3 Stand der Forschung

Zum Einsatz von GNNs zur approximativen Einordnung von Argumenten von AFs bezüglich bestimmter Semantiken existieren bisher vier Arbeiten, die wir als Fallbeispiele miteinander vergleichen möchten. Die vier bisherigen Forschungsarbeiten in diesem Feld unterscheiden sich sowohl in Architektur und Design der benutzten GNNs als auch den eingesetzten Datensätzen. Ein Vergleich der Leistung zwischen Modellen über Datensatzgrenzen hinweg ist wie im Abschnitt zu den Kennzahlen erwähnt nicht einfach. Trotzdem werden wir versuchen mit unserer Untersuchung einen Leistungsraum abzustecken, um zum einen die Ergebnisse der vier Arbeiten untereinander vergleichbar zu machen und zum anderen die Ergebnisse unserer Experimente mit denen dieser Arbeiten vergleichbar zu machen. Wir werden nun die für uns wichtigen Versuchsanordnungen und Ergebnisse der vier bisherigen Forschungsarbeiten vorstellen.

3.1 Fall A: `FM2` von KUHLMANN und THIMM (2019)

KUHLMANN und THIMM legen 2019 mit „Using Graph Convolutional Networks for Approximate Reasoning with Abstract Argumentation Frameworks: A Feasibility Study“([KT19]) den Grundstein des Feldes. Sie nutzen zwei verschiedene GCNs mit je zwei versteckten KIPF-WELLING-GCN-Schichten. Die GCNs unterscheiden sich nur in ihren Eingangsmerkmalen. Eines erhält dasselbe Eingangsmerkmal pro Argument (`FM1`), das andere die Anzahl der ein- und ausgehenden Kanten (`FM2`). Das behandelte Problem ist das der leichtgläubige Akzeptanz eines Arguments unter der bevorzugten Semantik DC_{pr}. Ihre Datengrundlage stellen selbstgenerierte AFs durch die sechs Generatoren in `probo` und `AFBenchGen`. Für Vergleichstests nutzen sie einen Teil der ICCMA-2017-AFs. Ihr Haupttrainingsdatensatz $A_{L(pbbg)}$[51] besteht aus je einer Untermenge von 100 AFs pro Typ. Sie nutzen zum Training noch zwei weitere AF-Datensätze. Der eine ist der in Bezug auf das Verhältnis von pa* zu pna* etwas ausgeglichenere Datensatz $A_{L(bal)}$, der 100 Barabási-Albert-AFs und nur je 10 der anderen 5 Typen enthält, so dass insgesamt 45,6% der Argumente pa* sind. Der andere ist der bezüglich pa* ausgeglichene Datensatz $A_{L(bal+)}$, der $A_{L(bal)}$ um 27 weitere Barabási-Albert-AFs augmentiert, so dass nun 50,6% der Argumente pa* sind. Der meist genutzte Abschlusstestdatensatz $A_{T(pbbg)}$ besteht aus je einer Untermenge von 20 AFs pro Typ. Als Ergänzung dient der Datensatz $A_{T(iccma)}$ mit 45 ICCMA-2017-AFs aus der Kategorie B.1 bis B.5.[52] Die selbstgenerierten AFs beinhal-

[51] Wir identifizieren jede Arbeit der bisherigen Forschung mit einem der Großbuchstaben von A bis D. Wir markieren wichtige Experimente mit fortlaufender Nummerierung. Wichtige AF-Datensätze erhalten stattdessen ein niedriggestelltes L, falls der Datensatz zum Lernen und Trainieren genutzt wird, und ein T, falls er zum Testen genutzt wird. Falls es mehr als einen relevanten Trainings- oder Testdatensatz gibt, markieren wir sie mit einem aussagekräftigen Kürzel. Wir ignorieren etwaige Validierungsdatensätzen.

[52] Im Abschnitt über die Auswahl der AFs, wird nicht gesondert erwähnt, ob einige dieser AFs einen ICCMA-2017-spezifischen AF-Typ haben (vgl. [KT19]: S. 29). Wir gehen aber im Weiteren davon

ten 100 bis 400 Argumenten bei einem Durchschnitt von ungefähr 250.[53] Die Argumente in $A_{T(pbbg)}$ sind zu 23% pa*, in den Untermengen schwankt der Wert zwischen 84% bei Barabási-Albert-AFs und 0,0011% bei Watts-Strogatz-AFs; in $A_{T(iccma)}$ sind gut 40% der Argumente pa*.

Im ersten Test A_1 werden `FM1` und `FM2` mit $A_{L(pbbg)}$ trainiert und dann an $A_{T(pbbg)}$ getestet. Die Architektur `FM1` zeigt hier mit einem MCC[54] von 0,0 keinen Lernerfolg, `FM2` erreicht einen MCC von bis zu 0,326. Das Merkmal der Zahl der ein- und ausgehenden Kanten zeigt eine große Wirkung auf die Leistung des GCNs. In allen Folgetests werden nur `FM2`-Designs benutzt. Im zweiten Test A_2 wird $A_{L(pbbg)}$ entlang der AF-Typen in Untermengen aufgeteilt, dann je ein Modell pro Untermenge trainiert und nur an AFs des passenden Typs aus $A_{T(pbbg)}$ getestet. Hier zeigen fast alle Modelle einen MCC von 0,0. Die Barabási-Albert-Graphen mit einem hohen Anteil an pa* Argumenten, trainieren Ja-Entschieder, die anderen Nein-Entscheider. Die Ausnahme bildet das Modell mit einem MCC von 0,204, das an Graphen des Grounded-Generators trainiert und getestet wurde. Insgesamt ist die Zahl der Trainingsgraphen pro Modell mit 100 klein. Auffällig ist, dass in den AFs des Grounded-Typs im Vergleich zu den AFs der anderen Typen der Anteil an pa* mit 31% am nächsten an einem ausgeglichenen Verhältnis steht. Dies ermöglicht vermutlich das erfolgreiche Lernen unter diesen Bedingungen. Der dritte Test A_3 wird mit verkleinerten Varianten von $A_{L(pbbg)}$ durchgeführt, in denen statt 100 nur 5, 10, 25, 50 oder 75 AFs pro Typ enthalten sind. Das Modell, das an nur insgesamt 30 Trainingsgraphen trainiert wurde, zeigt einen MCC von 0,0, aber schon das nächste an 60 AFs trainierte Modell einen von 0,298. Das heißt, schon 60 gemischte AFs ermöglichen eine bessere Erkennungsleistung als alle Einzeltest an den je 100 AFs des zweiten Tests. Je mehr AFs im Trainingsdatensatz sind, desto höher ist der MCC. Beim gesamten Datensatz $A_{L(pbbg)}$ beträgt er hier 0,335, was eine kleine Steigerung von +0,009 von A_1 zu A_3 bedeutet und einen erster Hinweis auf die Varianz der Ergebnisse liefert. Insgesamt verkleinert sich der Zugewinn: Zusätzliche AFs haben einen geringeren postiven Effekt auf den MCC als ihre Vorgänger. Das bessere Abschneiden wird durch eine bessere Erkennung der positiven Klasse erreicht. Die TPR steigt schneller wächst, als die TNR fällt. In dem vierten Test A_4 wird zum ersten Mal das aus iccma-AFs bestehende $A_{T(iccma)}$ genutzt und Lernparameter wie Epochenanzahl, Lernrate und *Dropout*-Rate varriiert. Der MCC-Wert von 0,340 gilt für 500 Epochen, eine Lernrate von 0,1 und eine *Dropout*-Rate von 0,05. Die MCC-Werte für andere Parameter unterscheiden sich kaum von ihm und reichen von 0,337 (-0,003) bei einer geringen Lernrate von 0,001 bis zu 0,342 (+0,002) bei einer auf 750 erhöhten Epochenzahl. Aber selbst eine geringere Epochenzahl von 250 hat auch einen sehr geringen positiven Effekt. Das Training mit dem etwas ausgeglichenen Datensatz $A_{L(bal)}$ zeigt

---

aus.

[53] So ist der Durchschnitt in $A_{L(pbbg)}$ $149.130/600 \approx 249$, in $A_{T(pbbg)}$ $30.603/120 \approx 255$ und in der Augmentierung des balanzierten Trainingsdatensatzes $A_{L(bal+)} \setminus A_{L(bal)}$ $7300/27 \approx 270$.

[54] KUHLMANN und THIMM notieren keine MCC-Werte, so dass wir sie wie im letzten Abschnitt zu den Kennzahlen erläutert aus TPR, TNR und ACC abgeleiten.

einen kleinen positiven Effekt mit einer MCC-Steigerung von +0,005 im Vergleich zu $A_{L(pbbg)}$ im dritten Test. Im fünften und letzten Test A_5 werden Modelle einmal mit dem ausgeglichenen Trainingsdatensatz $A_{L(bal+)}$ und einmal mit der Hälfte von $A_{L(pbbg)}$ (50 AFs jeden Typs) trainiert, dann wiederum an $A_{T(iccma)}$ getestet. Hier wird zum ersten Mal eine Übertragung getestet, da für die Modelle AFs der fünf ICCMA-2017-spezifischen AF-Typen unbekannt sind. Beide Modelle lernen Argumente aus $A_{T(iccma)}$ einzuordnen – wenn auch wesentlich schlechter – mit MCC-Werten von 0,146 für den Teil von $A_{L(pbbg)}$ und 0,156 für $A_{L(bal+)}$. Das mit $A_{L(bal+)}$ trainierte Modell erkennt pa* Argumente kaum mit einer TPR von 0,17, aber besser als das andere Modell mit einer TPR von 0,1. Dies wird durch ein umgekehrtes Verhältnis der TNR fast ausgeglichen. Das Resultat lässt kaum Rückschlusse auf den Effekt der Übertragung des Lernens auf ungesehene AF-Typen zu. Vergleichen wir dafür das Ergebnis des Modells trainiert an der Hälfte von $A_{L(pbbg)}$ aus A_5 mit dem Modell trainiert an der Hälfte von $A_{L(pbbg)}$ aus A_3. Es fällt auf, dass die TNR nur um 0,0097 von 0,9797 auf 0,97 fällt, die TPR bricht aber um 0,117 von 0,217 auf 0,1 ein. Der MCC fällt ähnlich stark von 0,335 auf 0,146. Da eine genauere Aufteilung des Ergebnisses fehlt, können wir keine belastbare Aussage darüber treffen, welchen Einfluss z. B. eine größere durchschnittliche Argumentenzahl pro AF oder eben die unbekannten AF-Typen im Testdatensatz haben. Diese Arbeit zeigt, dass mit einem aussagekräftigen Eingangsmerkmal versehene KIPF-WELLING-GCNs Argumente von unbekannten AFs bekannter AF-Typen in Maßen (MCC bis zu 0,342) korrekt einordnen können, wenn sie vorher mit genügend AFs dieser Typen trainiert wurden. Die Vergrößerung des Trainingdatensatzes hat einen positiven Einfluss, der mit steigender Zahl an AFs abnimmt. Er zeigt sich vor allem über eine steigende Genauigkeit der unterrepräsentierten Klasse. Variationen der Lernrate, Epochenzahl oder der *Dropout*-Rate haben hingegen nur geringen bis keinen Einfluss auf die Leistung. Balanzierungsmaßnahmen haben einen großen positiven Einfluss auf die Genauigkeit der unterrepräsentierten Klasse, aber nur einen begrenzten positiven Einfluss auf die Gesamterkennungsleistung gemessen im MCC-Wert. Inwieweit eine Übertragung des Gelernten auf unbekannte AF-Typen möglich ist, ist unklar.

3.2 Fall B: GCN mit Graphenrohresiduen von MALMQVIST (2020)

MALMQVIST, YUAN, NIGHTINGALE und MANANDHAR folgen 2020 mit „Determining the Acceptability of Abstract Arguments with Graph Convolutional Networks“ ([MYNM20]) dem Ansatz KUHLMANNS und THIMMS und vergleichen die Leistung von GCNs mit vier bis sechs KIPF-WELLING-Schichten bezüglich der leichtgläubigen und skeptischen Akzeptanz eines Arguments unter der bevorzugten Semantik DC_{pr} und DS_{pr}. Sie generieren Eingangsmerkmale über das *bag-of-words*-Verfahren `DeepWalk` ([PARS14]), welches Nachbarschaften von Argumenten über zufällige Schrittfolgen (engl.: *random walks*) approximiert und in einem niedrigdimensionalen Vektor komprimiert. Um die Leistung von tieferen KNNs zu stabilisieren, nutzen sie sogenannte Graphenrohresiduen (engl: *graph-raw residuals*). Die normierten, ur-

sprünglichen Eingangsmerkmale in H^0 werden zu den Ergebnissen der Konvolution jeder Schicht hinzuaddiert. Sie folgen damit [ZM19]. Graphenrohresiduen können auch als Auslassverbindungen verstanden werden, die die Neuronen aus der Eingabeschicht i mit jeweils einem Neuron aus h^2 bis h^m verbinden. Gegen eine mögliche Überanpassung durch die höhere Expressivität tiefer KNNs arbeiten sie mit einer hohen *Dropout*-Rate von 0,5. Wir nennen diese Architektur res-GCN mit dem Präfix ‚res' als Kürzel für `DeepWalk`-Graphenroh**res**iduen.[55] Als Suffix hängen wir für konkrete Designs die Anzahl an GCN-Schichten an. Die Autoren experimentieren auch mit Balanzierungstechniken für den Trainingsdatensatz, wozu hier ein Ausschluss von AFs vom Training gehört, die besonders wenige oder viele pa oder pa* Argumente haben,[56] sowie über Trainingsmasken, die garantieren, dass pro Epoche stets über die Einordnung einer ähnlichen Anzahl an pa und pna bzw. pa* und pna* Argumenten gelernt wird. Das mit den Balanzierungmaßnahmen trainierte Modell vergleichen sie mit den anderen Modellen ohne diese Balanzierungmaßnahmen. Leider werden beide Maßnahmen stets nur im Zusammenspiel und nicht einzeln getestet, so dass über die Effektivität der einzelnen Maßnahme kein Rückschluss möglich ist.

Ihre Datengrundlage bilden mit 900 AFs fast alle der ICCMA-2017-Graphen.[57] Zufällig wird ein Zehntel dieser AFs als Testdatensatz B_T ausgewählt und der Rest bildet den Trainingsdatensatz B_L. Das balanzierte GCN wird mit der Untermenge $B_{L(bal)}$ trainiert und mit der Untermenge $B_{T(bal)}$ getestet, aus denen vom Median stark abweichende AFs bezüglich des Anteils an pa bzw. pa* Argumenten ausgeschlossen wurden. Das inkludierte AF mit der höchsten Argumentenzahl hat 15.605 Argumente[58]; die höchste Angriffsanzahl beträgt 6.257.500[59]; im Durchschnitt ent-

[55] Diese Bezeichnung ist selbstverständlich nur für unsere Arbeit eindeutig. In einem anderen Zusammenhang kann er leicht mit z. B. ResGCN verwechselt werden ([PHvIP22]). Andererseits wird auch die Bezeichnung AGNN sowohl für die ‚*Argumentation Graph Neural Network*'-Architektur als auch für die ‚*Attention-based Graph Neural Network*'-Architektur benutzt. Erstere führen wir im folgenden Abschnitt ein.

[56] AFs, deren Anteil mehr als dreieinhalb Standardabweichungen vom Medianwert abweichen, werden ausgeschlossen (vgl. [MYNM20]: S. 51.).

[57] Der ICCMA-2017-Wettbewerb benutzte über verschiedene Problemkategorien und Schwierigkeitsklassen verteilt insgesamt 1050 AF-Instanzen, von denen einige mehrfach benutzt werden. Nach unserer Untersuchung gibt es 847 AFs mit einmaligem Namen. Für diese haben wir nur eine oberflächige Identitätsprüfung und keine Isomorphieprüfung durchgeführt, sind dabei aber auf drei weitere AFs gestoßen, die sich von anderen AFs nur im Dateinamen unterscheiden. Ansonsten sind sie von der Argumentbenennung bis zur Reihenfolge der Argumente identisch. Nach unserer oberflächlichen Prüfung bleiben höchstens 844 unterschiedliche AFs übrig. Wir wissen, dass weiterhin acht AFs wegen ihrer Größe ausgeschlossen wurden (siehe auch folgende Fußnote), was die Zahl auf höchstens 836 unterschiedliche AFs senkt. Dies bedeutet, dass mindestens 64 Instanzen der AFs des genutzten Datensatzes identisch zu anderen Instanzen sind. Deshalb ist es möglich, dass mit demselben AF trainiert und getestet wurde, was die Aussagekraft der Ergebnisse etwas schmälert.

[58] Das AF Pace-suburban-bus-service_20121002_0305.gml.80 hat diese Knotenanzahl. Die wegen ihrer Größe ausgeschlossen AFs sind dementsprechend alle vom Typ AdmBuster mit 20.000, 50.000, 100.000, 200.000, 500.000, 1.000.000, 1.500.000 und 2.000.000 Argumenten.

[59] SemBuster_7500 hat diese Angriffsanzahl. Kein AF hat mehr Angriffe, also wird aufgrund der An-

hält ein AF 1595 Argumente und 187.542 Angriffe. Die Argumente von B_T sind nur zu ungefähr 3,6%pa* und sogar nur 1,2%pa.

Das erste Experiment B_1 vergleicht die Leistung der Designs in DC_{pr}. Die MCC-Werte[60] beim dem Vergleich von `res-GCN4`, `res-GCN5` und `res-GCN6` bezüglich leichtgläubiger Akzeptanz, liegen bei 0,413, 0,422 und 0,4, mit `res-GCN5` knapp in Führung.[61] Die TPR-Werte liegen um 0,7 herum, die TNR zwischen 0,92 und 0,94. Leider muss die in der Arbeit über Inklusion in einer gemeinsamen Tabelle angedeutete Vergleichbarkeit mit den `FM2`-Ergebnissen eingeschränkt werden, da die Daten direkt dem fünften Test A_5 entnommen wurden, statt an ihrem eigenen Datensatz trainiert und getestet zu werden. Denn die einsetzten Datensätze unterscheiden sich stark. So sind die AFs in B_T mit durchschnittlich 1.595 Argumenten wesentlich größer als die in $A_{T(iccma)}$ mit ungefähr 412.[62] Weiterhin sind die enthaltenen Argumente wesentlich seltener in pa* (3,6% statt 40%). Aber alle diese Einschränkungen gehen eher zu Lasten der Leistungswerte der res-GCNs. Größere AFs sind generell schwerer einzuordnen, genauso wie Datensätze mit größeren Unterschieden in der Klassenverteilung. Das bedeutet, dass wir die Leistung von res-GCN-Designs in diesem Vergleich eher unterschätzen. Ähnliche Hindernisse bezüglich der Vergleichbarkeit finden wir beim Vergleich zwischen den normal trainierten und gestesteten Modellen und den balanziert trainiert und getesteten Modellen. Für eine direkte Vergleichbarkeit hätte das balanziert trainierte Modell auch an B_T getestet werden müssen, da sich B_T und $B_{T(bal)}$ (mit Absicht) stark voneinander unterscheiden. So sind in $B_{T(bal)}$ 50,5% der Argumente pa* statt 3,6% in B_T. Das balanziert trainierte Modell erreicht mit einem MCC von 0,636 einen guten Wert für eine GCN-Architektur und zeigt dabei eine hohe TPR von 91,2% und eher niedrige ACC von 81,2%. Da die Vergleichbarkeit eingeschränkt ist, können wir das Ergebnis nur für sich betrachten. Es zeigt, dass GCNs mit einem ausgeglichenen Datensatz bezüglich pa* die Einordnung von pa* Argumenten genauso gut oder besser lernen können wie die pna* Einordnung. Außerdem erlauben es ausgeglichenere Datensätze zusammen mit dem balanzierten Training durch die Trainingsmasken, dass res-GCN-Modelle statt niedriger insgesamt mittelhohe MCC-Werte erreichen können. Beim zweiten Experiment B_2 mit skeptischer Akzeptanz zeigen `res-GCN4`, `res-GCN5` und `res-GCN6` wesentlich niedrigere MCC-Werte von 0,1326, 0,1265 und 0,1259[63] mit `res-GCN4` in Führung.[64] Die Ergebnisse des balanzierten GCNs

griffsanzahl kein AF ausgeschlossen. Im Aufsatz wird von 6.250.500 Angriffen als höchster Angriffsanzahl eines Graphen gesprochen ([MYNM20]: S. 52). Allerdings weist kein AF diese Anzahl auf und der nächstkleinere Wert ist 5.378.385. Wir nehmen deshalb an, dass es sich um einen Zahlendreher handelt und SemBuster_7500 gemeint ist.

[60] Auch hier leiten wir die MCC-Werte aus TPR, TNR und ACC ab.

[61] In diesen Daten zeigt die Gesamtgenauigkeit ACC hingegen knapp `res-GCN4` mit 92,68% gegenüber 92,26% vorne.

[62] Wir haben keine exakten Zahlen für $A_{T(iccma)}$, aber 412 ist die Durchschnittsgröße der AFs in ICCMA-2017 B.1-5, von denen $A_{T(iccma)}$ eine Auswahl bildet.

[63] Wir runden hier ausnahmsweise auf vier Stellen nach dem Komma, um die absteigende Tendenz nicht durch Rundungsfehler zu verdecken.

[64] Hier zeigt ACC hingegen knapp `res-GCN6` mit 96,24% gegenüber 96,21% vorne.

müssen wir leider ignorieren, da sie widersprüchlich sind.[65]

Insgesamt hat die Arbeit gezeigt, dass mit der res-GCN-Architektur tiefere GCNs als `FM2` mit einer höheren Erkennungsleistung möglich sind. Sie degenerieren dabei nicht zu Ja- oder Nein-Entscheidern und die Erhöhung von vier auf sechs Schichten zeigt nur einen minimalen Leistungsabfall. Für den Ausschnitts des ICCMA-2017-Datensatzes erreichen die verschiedenen Modelle je nach Problem und je nach Balanzierung der Trainings- und Testdaten unterschiedliche MCC-Werte von im schlechtesten Fall 0,1259 bis zu 0,636. Es bleibt festzuhalten, dass für die Vergleichbarkeit die Nutzung derselben Testdatenmenge oder zumindest die Nutzung einer Menge mit sehr ähnlichen Eigenschaften, falls die Testdatenmenge nicht zur Hand ist, wichtig ist. So bleiben alleine durch die unterschiedlichen Testdatenmengen im pa*-Test die Fragen übrig, welchen Beitrag an dem guten Abschneiden des balanzierten GCNs der ausgeglichene Testdatensatz $B_{T(bal)}$ hatte und welchen Beitrag der ausgeglichene Trainingsdatensatz $B_{L(bal)}$ hatte. Offen bleibt auch, welchen Einfluss ein Training mit $B_{L(bal)}$ auf ein Testen mit B_T hätte, welchen Einfluss ein Training mit B_L auf ein Testen mit $B_{T(bal)}$ hätte, oder welchen Einfluss überhaupt das Ausbalanzieren der Trainingsmasken bei einem sonst normalen Training mit B_L und einem Test mit B_T gehabt hätten. Weiterhin wurde der Einfluss der `DeepWalk`-Nachbarschaftsaggregationen als Eingangsmerkmale (z. B. im Gegensatz zur Zahl der ein- und ausgehenden Kanten), sowie als Residuen (z. B. im Gegensatz zu keinen Residuen) auf die Leistung nicht getrennt untersucht. Wir halten als letzte Beobachtung fest: Wie `FM2` zeigen auch res-GCN-Design niedrigere MCC-Werte, wenn die die Klassenverteilung unausgeglichener ist.

3.3 Fall C: AGNN von CRAANDIJK und BEX (2020)

CRAANDIJK und BEX erreichten mit einer spezialisierten Nachrichtenübermittlungsarchitektur statt klassischen GCN-Konvolutionen in „Deep Learning for Abstract Argumentation Semantics“([CB20]) eine sehr hohe Klassifikationsgenauigkeit für kleine AFs. Ihr AGNN (engl.: *argumentation graph neural network*) ist ein minimal heterogenes GNN. Zwei unterschiedliche MLPs stellen in jedem Übermittlungsschritt Nachrichten sowohl in als auch entgegen der Angriffsrichtung zusammen. Die Aktualisierung der Merkmale wird per RNN[66] realisiert, das aus dem aktuellen Zustand des RNNs, den summierten Nachrichten und dem Ausgangsmerkmal jedes Arguments den neuen Merkmalswert des Arguments und den neuen Zustand des RNNs berechnet. Die Auslesefunktion ist ein weiterer MLP, der einem Merkmalswert eines Argument einen Logitwert zuordnet. Jedem AF wird ein zufälliger Wert

[65] MALMQVIST et al. geben eine ACC von 97,15% bei den Teilgenauigkeiten TPR von 46,35% und TNR von 94,39% an. Letztere sind beide kleiner als die Gesamtgenauigkeit, was bei der Nutzung derselben TP-, FN-, TN- und FP-Werte nicht möglich ist.

[66] RNNs (engl.: *recurrent neural network*) sind MLPs mit Rückkopplung (engl.: *feedback connection*), in denen die Verarbeitung eines Signals an das Ergebnis (oder Zwischenzustände) der Verarbeitung des vorherigen Signals rückgekoppelt ist (vgl. mehr zu RNNs und ihren verschiedenen Ausformungen: [GBC16]: S. 378ff.).

zugewiesen und dieser wird wie bei FM1 allen Argumenten des AFs zu Beginn als Eingangsmerkmal mitgegeben. CRAANDIJK und BEX prüfen die Erkennungsleistung ihres Modells für leichtgläubige und skeptische Akzeptanz in der bevorzugten, grundierten, stabilen und vollständigen Semantik. Ihre Datengrundlage stellen selbstgenerierte AFs durch die sechs Generatoren in probo und AFBenchGen. Der Trainingsdatensatz C_L besteht aus einer Million kleinen AFs mit 5 bis 25 Argumenten. Die Testdatensätzen C_{T25} bis C_{T200} bestehen aus je 1000 AFs mit 25, 50, 100 und 200 Argumenten.[67]

In der ersten Testreihe C_1 vergleichen die Autoren die Leistung des AGNNs mit einem KIPF-WELLING-GCN[68] und dem FM2-GCN bezüglich leichtgläubiger und skeptischer Akzeptanz unter der grundierten, bevorzugten, stabilen und vollständigen Semantik. Sie nutzen dabei den Testdatensatz C_{T25}. Für diese Testreihe notieren sie die Ergebnisse des AGGNs nach 32 Nachrichtenübermittlungsschritten.[69] Das AGNN zeigt herausragende bis perfekte MCC-Werte von 0,997 bis 1,0 (Ø[70]: 0,999), das einfache GCN erreicht hingegen nur Werte von 0,16 bis 0,39 (Ø: 0,225) und FM2 erreicht nur Werte von immerhin 0,54 bis 0,64 (Ø: 0,586). Das AGNN lernt die beiden Akzeptanzen der grundierten und vollständigen Semantik perfekt und für die bevorzugte und stabile Semantik lernt es die leichtgläubige Akzeptanz ein wenig besser als die skeptische. Für FM2 ist auch die grundierte und vollständige Extension leichter zu erlernen, allerdings lernt es die skeptische Akzeptanz etwas besser. Das KIPF-WELLING-GCN lernt die stabile Extension am besten und die leichtgläubige Akzeptanz wesentlich besser ($Ø_{cred}$: 0,273, $Ø_{scep}$: 0,178). In der zweiten Testreihe C_2 wird die Leistung eines trainierten AGNN-Modell an allen Testdatensätzen mit verschieden großen AFs bezüglich DC_{pr} getestet. Dies wird nicht wie sonst nach 32 Nachrichtenübermittlungsschritten gestoppt, sondern weitergeführt und Zwischenwerte aufgezeichnet. Schon nach zwei Schritten zeigt das AGNN im schlechtesten Falle einen höheren MCC-Wert von 0,66[71] bei C_{T100} als FM2 im ersten Test bei C_{T25}. Nach 32 Schritten ist der schlechteste Fall ein MCC von 0,85 für C_{T100} und C_{T200} und 0,99+ und 0,96 für C_{T25} und C_{T50}. Nach 1024 Schritten erreichen die C_{T25} und C_{T50} einen MCC von nahe 1,0, C_{T100} erreicht einen MCC von 0,92 und die der C_{T200} erreicht einen von 0,89.

[67] Die genaue Aufteilung dieser Datensätze in Typen wird in der Arbeit nicht erwähnt, wir gehen von relativ gleichen Anteilen aus.

[68] Die Autoren beschreiben dieses KNN nicht genauer. Wir gehen von einem zweischichtigem GCN aus, der als Eingabemerkmale ähnlich wie der AGNN oder FM1 eine Zufallszahl pro AF erhält.

[69] Damit gehen Informationen aus der 32-Hop-Nachbarschaft des ungerichteten AFs (wegen der Nachrichtenübermittlung in und entgegen der Angriffsrichtung) in die Berechnung eines Merkmalswertes ein, statt wie bei FM2 Informationen aus der 2-Hop-Nachbarschaft oder wie bei res-GCN6 aus der 6-Hop-Nachbarschaft.

[70] Die Durchschnitte sollten nicht direkt mit anderen MCC-Werten verglichen werden, sondern bringen nur eine verkürzte – wenn auch grobe – Orientierung über die Verteilung der Werte.

[71] Die folgenden Datenwerte wurden aus der Abb. 2 ([CB20]: S. 1671) mithilfe des externen Programm WebPlotDigitizer (https://automeris.io/WebPlotDigitizer/) händisch abgeleitet. Damit approximieren wir die Original-MCC-Daten, die damit um bis zu 0,03 höher oder niedriger liegen können.

Insgesamt hat diese Arbeit gezeigt, dass mit der AGNN-Architektur herausragende Vorhersagen zumindest für kleine Graphen mit bis zu 50 Argumente zu treffen sind und dass für mittelgroße AFs bis 200 Knoten immer noch äußerst gute Vorhersagen möglich sind. Letztere lassen sich noch durch mehr Nachrichtenübermittlungsschritte steigern. Die Leistung für `FM2` für die 25-Argumente-AFs ist mit einem MCC von bis zu 0,64 überraschend gut, da es den besten Wert in Fall A von KUHLMANN und THIMM fast verdoppelt. Leider wurden die Graphen mit mehr Argumenten nicht an `FM2` getestet, so dass es unklar bleibt, ob die Leistung eines `FM2`-Modells mit steigender Argumentenzahl ähnlich wie die eines AGNN-Modells oder schneller oder sogar langsamer abnimmt. Letzteres würde bedeuten, dass der große Vorteil von AGNNs bei sehr kleinen AFs bei größeren AFs schrumpfen könnte, oder viele rechenintensive Nachrichtenübermittlungsschritte erforderlich machen könnte.

3.4 Fall D: AGNN und AF-Typen von KUHLMANN, WUJEK und THIMM (2022)

In dem Aufsatz „On the Impact of Data Selection when Applying Machine Learning in Abstract Argumentation“ bauen KUHLMANN, WUJEK und THIMM auf dem AGNN-Ansatz auf ([KWT22]). Sie vollziehen den Prozess von CRAANDIJK und BEX nach und untersuchen zusätzlich wie AFs des neuen `KWT`-Generators erkannt werden, bzw. das Training beeinflussen, sowie wie unterschiedlich gut die trainierten Modelle die verschiedenen AF-Typen des ICCMA-2017-Datensatzes einordnen können. Für unsere Arbeit ist besonders dieses Auffächern der Leistung nach Typen relevant. Die Autor*innen behandeln nur DS_{pr}. Die Datengrundlage des Trainings ist $D_{L(pbbg)}$ mit 100.000 selbstgenerierte Graphen durch die sechs Generatoren in `probo` und `AFBenchGen` mit je 5 bis 25 Knoten, sowie eine davon unabhängige Trainingsmenge $D_{L(kwt)}$ mit 1000 kwt-Graphen mit 151 Argumenten. Für die Abschlusstests werden drei Datensätze genutzt: $D_{T(pbbg)}$ mit 1000 AFs der sechs Generatoren in `probo` und `AFBenchGen` mit je 25 Argumenten (von allen Argumenten sind 28,3% pa), $D_{T(kwt)}$ mit 1000 kwt-Graphen mit je 151 Argumenten (46% aller Argumente sind pa) und $D_{T(iccma)}$ mit 450 der ICCMA-2017-Graphen mit durchschnittlich 649 Argumenten (11% aller Argumente sind pa).

In der Testreihe D_1 vergleichen die Autor*innen die MCC-Werte von zwölf Konfigurationen. Je ein AGGN-Modell und ein `FM2`-Modell werden an den beiden Trainingsmengen trainiert. Diese vier Modelle werden dann an jeder der drei Testmengen getestet. Damit ist ein Vergleich der Qualität der Übertragung des Trainings auf nie gesehene AF-Typen möglich. Erneut schneidet das AGGN wesentlich besser als `FM2` ab. Sind Generatoren von Trainings- und Testmenge identisch, erreicht der AGGN einen MCC von 0,962 ($D_{L(pbbg)}$, $D_{T(pbbg)}$) und 0,927 ($D_{L(kwt)}$, $D_{T(kwt)}$), der `FM2` erreicht hingegen nur 0,558 ($D_{L(pbbg)}$, $D_{T(pbbg)}$) und 0,364 ($D_{L(kwt)}$, $D_{T(kwt)}$). Der MCC-Wert von 0,962 ist überraschend niedrig im Vergleich mit den 0,997, die

CRAANDIJK und BEX unter ähnlichen Bedingungen[72] erreichten. Die Differenz zu einem perfekten MCC von 1,0 ist hier in D mit 0,038 zu C mit 0,003 mehr als zehnmal so groß. Die sechsmal größeren kwt-Graphen sind für beide GNN-Typen schlechter einzuschätzen, wobei der AGNN hier mit 0,927 zu 0,85 wesentlich besser abschneidet als bei CRAANDIJK und BEX im Vergleichstest mit 100- und 200-Argumente-AFs, die mit `probo` und `AFBenchGen` generiert wurden.[73] Das Training mit kwt-AFs bereitet `FM2` kaum auf den Testdatensatz ohne kwt-AFs vor. Mit 0,078 für $D_{T(pbbg)}$ und 0,055 für $D_{T(iccma)}$ sind die Werte kaum besser als die `FM1s` in A_1, was wie dort in der sehr schlechten Erkennung der positiven Klasse begründet ist; die TPR beträgt nur 0,03 und 0,016. Beim Training mit den anderen AFs gelingt `FM2` die Übertragung etwas besser. So liegt der MCC beim Training mit $D_{L(pbbg)}$ und Testen mit $D_{T(kwt)}$ mit 0,359 fast genau so hoch wie beim Training mit den kwt-AFs mit $D_{L(kwt)}$ mit 0,364: Auf das Lösen von kwt-AFs werden `FM2`-Modelle von kwt-AFs selbst kaum besser als von AFs der sechs Generatoren vorbereitet.[74]. Die zum Teil unbekannten AF-Typen angehörigen, noch einmal wesentlich größeren ICCMA-AFs in $D_{T(iccma)}$ werden im Vergleich zu den anderen beiden Testdatensätzen schlecht bis moderat gut erkannt. Wieder ist der AGNN weit in Führung mit einem MCC-Wert von 0,507 zu 0,211 beim Training mit $D_{L(pbbg)}$ und von 0,250 zu 0,055 beim Training mit $D_{L(kwt)}$.

Anschließend schlüsseln die Autor*innen die Ergebnisse an den beiden Trainingsdatensätzen trainierten AGNN-Modelle getestet an $D_{T(iccma)}$ für alle elf AF-Typen des ICCMA-2017-Datensatzes im Experiment[75] D_2 auf. Die MCC-Werte sind zunächst überraschend breit gestreut: Die des an $D_{L(pbbg)}$ trainierten Modells reichen von -0,110 (AdmBuster) bis 0,946 (Barabási-Albert), die des an $D_{L(kwt)}$ trainierten Modells von -0,454 (AdmBuster) bis 0,680 (ABA2AF). Es gibt aber keine eindeutige Korrelation zwischen der Inklusion eines AF-Typs im Training und seinen MCC-Werten. Die in $D_{L(pbbg)}$ inkludierten Typen Erdős-Renyi, SCC und Watts-Strogatz zeigen beim Test mit $D_{T(iccma)}$ schlechte bis sehr schlechte Werte von -0,001, 0,135 und 0,262, Stable- und Grounded-AFs hingegen akzeptable Werte von 0,463 und 0,690 und nur die AFs vom Typ Barabási-Albert den hervorragenden Wert von 0,946. Der AGNN konnte also nicht von seinem Training an sehr kleinen Erdős-Renyi-AFs auf die wesentlich größeren Erdős-Renyi-AFs des ICCMA-2017-Datensatzes profitieren. Gleichzeitig werden aber unbekannte AF-Typen wie Traffic und ABA2AF mit 0,789 und 0,804 überraschend gut erkannt. Das schlechte Abschneiden der sehr regelmäßigen AdmBuster-AFs mit -0,110 für $D_{L(pbbg)}$ und -0,454 für

[72] Wir vermuten hier, dass die Parameter der AF-Erstellung sich unterscheiden. Der größte für uns bezifferbare Unterschied ist der im ersten AGNN-Aufsatz zehnmal größere Trainingsdatensatz Einen MCC von rund 0,96 zeigen dort die wesentlich komplexeren AFs mit je 50 Argumenten.

[73] Auf die Komplexität der Einordnung von kwt-Graphen gehen wir am Ende des Abschnittes 5.6 nocheinmal ein.

[74] Wir erreichen in einem vergleichbaren Experiment für eine `FM2`-ähnliche Architektur einen besseren MCC-Wert. Siehe derselbe Abschnitt 5.6.

[75] Wir zählen die Auffächerung hier als neues Experiment, um die Referenzierung zu erleichtern, obwohl es sich genau genommen nur um eine andere Ergebnisdatenanalyse von D_1 handelt.

$D_{L(kwt)}$ liegt vermutlich in ihrem großen Durchmesser begründet, der Lösungsversuche über Nachrichtenübermittlung erschwert, wenn das Modell die Struktur nicht kennt.[76] In AdmBuster-AFs ist nur ein Argument zu Beginn sicher pa und nur Argumente in höchsten 32-Hop-Entfernung von diesem können in AGNNs ihre Zugehörigkeit sicher einschätzen. Das sind in AdmBuster-AFs die große Minderheit von 65 Argumenten.[77] Für eine Lösung im Sinne des AGNN-Modells, wären bei einem AdmBuster-AF mit acht Argumenten vier Nachrichtenübermittlungsschritte nötig, bei 1000 Knoten schon 500 Nachrichtenübermittlungsschritte. Die Ergebnisse des an $D_{L(kwt)}$ trainierten Modells unterscheiden sich im Großen und Ganzen nur darin, dass AF-Typen, die schon mit $D_{L(pbbg)}$ schlecht gelernt werden, mit $D_{L(kwt)}$ sehr schlecht gelernt werden und dass ABA2AF, Barabási-Albert und Traffic unter dem Wechsel von $D_{L(pbbg)}$ zu $D_{L(kwt)}$ weniger leiden als die anderen Typen. Wir können spekulieren, dass kwt-AFs diesen eher als den anderen Typen ähneln oder sich AFs dieser Typen im allgemeinen gut zum Lösen durch AGNNs eignen. Insgesamt zeigen diese Ergebnisse die ersten Einschränkungen in der Leistungsfähigkeit von AGNNs. Leider fehlt der Vergleich mit dem `FM2`-Modell, um herauszufinden, inwiefern diese Probleme in mehr als einer GNN-Architektur auftreten oder AGNN-spezifisch sind. In einem weiteren Test D_3 nutzen die Autor*innen einen naiven Klassifizierer. Falls die Anzahl der eingehenden oder ausgehenden Angriffe eines Argumentes kleiner als die Anzahlen für ein durchnittliches pa Argument sind, ordnet der Klassifizierer das Argument als pa ein. Sie erreichen damit beim Training mit $D_{L(pbbg)}$ und Testen mit $D_{T(pbbg)}$ einen MCC von 0,396, beim Training mit $D_{L(pbbg)}$ und Testen mit $D_{T(iccma)}$ 0,364 und beim Training mit $D_{L(kwt)}$ und Testen mit $D_{T(kwt)}$ 0,739. Die letzten beiden MCC-Werte schlagen die entsprechenden MCC-Werte von `FM2`. Mit anderen Methoden des maschinellen Lernens, die nicht auf einer Graphstruktur aufbauen und als Eingaben nur die Zahl der ein- und ausgehenden Kanten bekommen, erreichen Autor*innen beim Training mit $D_{L(pbbg)}$ und Testen mit $D_{T(pbbg)}$ einen MCC von bis zu 0,630 (+0,234 im Vergleich zum naiven Klassifizierer) und in den anderen beiden Tests 0,399 (+0,035) und hervorragende 0,924 (+0,185). Die Autor*innen fragen dann bezüglich des Merkmals der Anzahl der eingehenden und ausgehenden Angriffe: „[...]whether (and if so, in which manner) such a dominant feature influences a neural network's training process"([KWT22]: S.234).

Insgesamt zeigt die Arbeit, dass ein AGNN sehr gute bis herausragende Ergeb-

[76] Wir trainierten unser `FM2`-Pendant `GCN2LIN0` an unseren pbbg-AFs und testeten es am selben iccma-Datensatz und erreichen unter der idealen Semantik einen etwas höheren, weiterhin sehr schlechten MCC-Wert von fast 0,0 statt -0,110.

[77] Ein Blick auf die regelmäßige Struktur in [CP] zeigt, dass selbst für AGNNs und ihre bidirektionale Verarbeitung von Angriffen nur die Zugehörigkeit eines Arguments zu Beginn feststeht und pro Hop je zwei sichere Einordnungen dazukommen. Statt einer Lösung über die Weiterleitung der sicheren Zugehörigkeit des einen Argumentes muss ein GNN für einen höheren MCC-Wert eher den Typ des AFs als Grapheneigenschaft erkennen und dann die Argumente entsprechend einzuordnen. Dafür muss er aber AdmBuster-AFs kennen. Diesen Vorgang beobachten wir vermutlich in unserem Experiment $E(iccma)$ (siehe Abschnitt 5.5).

nisse an kleinen und mittelgroßen AFs aus AF-Typen zeigt, mit denen es trainiert wurde, aber kein robustes Verhalten bei AFs unbekannter Typen oder anderer Größe zeigt. Aber wie die Autor*innen im Fazit selber anmerken: Für die Lösung kleiner AFs werden keine approximativen Verfahren benötigt. Und das Lernen an sehr kleinen AFs ist nicht sicher auf größere AFs übertragbar. So zeigen die Tests an größeren iccma-AFs ganz unterschiedliche Ergebnisse selbst bei Typen, mit denen das AGNN trainiert wurde. In unserer Analyse der Datensätze in Abschnit XXXX werden wir ein Augenmerk auf diese Tatsache haben. Bei manchen unbekannten AF-Typen wie AdmBuster sind große Probleme ersichtlich, bei anderen wie ABA2AF hingegen gelingt die Übertragung und die Einordnung der Argumente ist leistungsstark. Zudem ist die Zahl der ein- und ausgehenden Kanten ein so starkes Merkmal für die Zugehörigkeit zu pa, dass es möglicherweise das Lernen von GNNs stark beeinflussen kann.

3.5 Ausschluss: EGNN von CRAANDIJK und BEX (2022)

In einer weiteren Arbeit beschreiben CRAANDIJK und BEX die neue KNN-Architektur EGNN (engl.: *enforcement graph neural network*, [CB22]), die sich in ihrem Ansatz stark von den anderen hier besprochenen Fällen unterscheidet, weshalb wir sie nicht als Vergleichspunkt und Fallbeispiel mitaufnehmen. Zum einen untersuchen die Autoren nicht das übliche Problem der Knotenklassifikation: Ist ein Argument a Teil einer, bzw. jeder σ-Extension? Sondern sie untersuchen das Problem der Statuserzwingung (engl.:*status enforcement*). In der Statuserzwingung wird ein $AF = (A, R)$ sowie ein $S \subseteq A$ mit $S = P \cup N$ und $P \cap N = \emptyset$ gegeben. Das Problem lautet hier: Wie muss R zu R' verändert werden, dass alle Argumente in P bezüglich einer Semantik σ DC_σ bzw. DS_σ sind und für die Argumente aus N das Gegenteil gilt, und dabei die Anzahl der Veränderungen $|R \setminus R'| + |R' \setminus R|$ minimal wird? Damit ist selbstverständlich kein Ergebnis aus [CB22] mit denen der anderen Arbeiten oder unseren Ergebnissen vergleichbar. Zum anderen nutzt ihr KNN – wenn wir ihre Beschreibung beim Wort nehmen – Schichten, die eher an lineare als an GNN-Schichten erinnern. So sind von den Angriffen zwischen Argumenten nur noch die Informationen in dem KNN enthalten, denn „[o]ur model takes an AF and maps it to a **fully connected** graph where nodes and edges have a vectorial representation denoting which nodes represent an argument that should be enforced and which edges represent an existing attack relation“ ([CB22]: S. 5576, Hervorh. d. Verf.). Bei der Nachrichtenzusammenstellung in einem EGGN wird die Nachricht von einem Knoten j an einen Knoten i über die aktuellen Knotenmerkmale der beiden Knoten und über die Kantenmerkmalen der Kante von i nach j und der Kante von j nach i gebildet. Dies sind ein einem vollständigen Graphen aber alle Knoten und alle Kanten. Für ein Argument i bilden damit alle anderen Argumente in A seine Nachbarschaft. Damit nutzt ein EGNN eine spezielle Schicht, die Eigenschaften von GNN- und linearen Schichten enthält. Die Struktur von einem Knoten pro Argument und dem Aktualieren seines Merkmalswertes über Nachrichten seiner

Nachbarn finden wie beim GNN-Nachrichtenübertragen wieder. Aber anstatt die Kantenstruktur eines AFs selbst zur Nachrichtenübermittlung zu benutzen, wird sie nur zur Qualifizierung der Kanten eines vollständigen Graphens genutzt und wie in linearen Schichten fließt die Information des aktuellen Zustands eines Knotens im aktuellen Schritt (der aktuellen linearen Schicht) zu allen Knoten für den nächsten Schritt (in die nächste lineare Schicht). Damit halten wir EGNNs für KNNs, die mit Graphenstrukturen arbeiten, und nicht für GNNs, die direkt auf diesen Strukturen arbeiten. Während des Trainingsprozesses bleiben die Kantenmerkmale dabei stets gleich und nur die Knotenmerkmale werden aufgrund der eingehenden Nachrichten, des aktuellen Merkmalswerts und des ursprünglichen Merkmalswerts (als *skip connection*) aktualisiert. Weiter schreiben die Autoren: „[a]fter a number of message passing steps the node and edge representations are mapped to the predicted Q-value **for each edge**“ (ebd., Hervorh. d. Verf.). Die Bewertung für jede Kante, ob sie zum Schluss Teil von R' wird, übernimmt wie im AGNN ein nachgeschaltetes KNN, dass dies über die aktualisierten Knotenmerkmale und die ursprünglichen Kantenmerkmale entscheidet. Insgesamt ist EGNN eine interessante Architektur und die in [CB22] berichtete Leistung im Vergleich zu klassischen Lösern und einfachen GCNs beeindruckend, aber für unsere Problemstellung ist dieser Ansatz nicht relevant.

3.6 Zusammenfassung des Standes der Forschung

In der Tabelle 4 haben wir die Zusammensetzung der benutzten Trainings- und Testdatensätze der Fälle A bis D zusammengestellt. Für jede relevante Eigenschaft der Datensätze findet sich eine große Bandbreite an Werten. So schwankt die Anzahl der AFs pro Trainingssatz zwischen 200 in $A_{L(bal)}$ bis 1.000.000 in C_L, und die Summe aller Argumente zwischen ungefähr 50.000 in $A_{L(bal)}$ bis 15.000.000 in C_L. Eine höhere Anzahl an Argumenten erleichtert das Lernen des Modells auf Kosten der Trainingsdauer (mit abnehmenden Effekt). Die durchschnittliche Anzahl an Argumenten pro AF schwankt in einem engeren Bereich zwischen 15 in C_L mit kleinen pbbg-AFs bis zu 1595 in B_L mit den größeren iccma-AFs. Mit steigender Argumentenzahl steigt die Wahrscheinlichkeit, dass die Zugehörigkeit eines Arguments durch die Zugehörigkeit eines entfernteren Arguments bestimmt wird, und damit steigt die Komplexität der Nachrichtenübermittlung (weshalb größere AFs im Experiment C_2 schlechtere Erkennungswerte bei gleicher Anzahl an Nachrichtenübermittlungsschritten zeigen als kleinere AFs). Der Anteil der pa, bzw. pa* Argumente zeigt auch eine breite Bandbreite, wobei der Anteil selten über 50% steigt. Für die Testdatensätze gilt Ähnliches, wobei die Bandbreite der Anzahl der AFs und der Anzahl an Argumenten insgesamt pro Datensatz immer noch vorhanden aber kleiner ist. Aber nur die pa und besonders die pa* Anteile der Testdatensätze des Falls B zeigen eine sehr hohe Unausgeglichenheit, die die direkte Vergleichbarkeit der MCC-Werte von DC_{pr} und DS_{pr} mit anderen Werten in Frage stellt.

[78] Der balanzierte Datensatz enstand, indem aus B_L AFs, deren Anteil an pa Argumenten mehr als dreieinhalb Standardabweichungen vom Median abweichen, ausgeschlossen wurden.

Name	#AFs	Typen	Ø $\lvert A\rvert$	Summe aller $\lvert A\rvert$	%pa*	%pa	Anmerkungen
$A_{L(pbbg)}$	600	pbbg	249	149.130	23**		100 AFs pro Typ
$A_{L(bal)}$	200	pbbg	249*	50.000*	45,6		100 BA, 20 × andere
$A_{L(bal+)}$	227	pbbg	252*	57.300*	50,6		127 BA, 20 × andere
B_L	810	ICCMA	1595*	1.291.950*	3,6**	1,2**	
$B_{L(bal)}$		ICCMA			51**		Ausreißerausschluss[78]
C_L	1.000.000	pbbg	15*	15.000.000*	32**		
$D_{L(pbbg)}$	100.000	pbbg	19	1.898.327		31,4	
$D_{L(kwt)}$	1000	kwt	151	151.000		46	
$A_{T(pbbg)}$	60	pbbg	255	30.603	23*		
$A_{T(iccma)}$	45	ICCMA			40*		
B_T	90	ICCMA	1595*	143.550*	3,6*	1,2*	
$B_{T(bal)}$		ICCMA			51*		Ausreißerausschluss
C_{T25}	1000	pbbg	25	25.000	32	23,6	
C_{T50}	1000	pbbg	50	50.000			
C_{T100}	1000	pbbg	100	100.000			
C_{T200}	1000	pbbg	200	200.000			
$D_{T(pbbg)}$	1000	pbbg	25	25.000		28,3	
$D_{T(kwt)}$	1000	kwt	151	151.000		45,8	
$D_{T(iccma)}$	450	ICCMA	649	292.221		10,8	

Tabelle 4: Überblick über die von der bisherigen Forschung genutzten Trainings- und Testdatensätze. Mit einem Stern markierte Werte sind fundierte Schätzungen mittels der veröffentlichen Kennzahlen oder anderer publizierter Hinweise. In anderen Fällen haben wir die Werte für die Trainingsdaten grob über die Werte der Testdaten geschätzt. Diese Einträge markieren wir mit zwei Sternen.

Vergleichen wir nun alle Ergebnisse nocheinmal miteinander über Fallgrenzen hinweg (siehe zur Übersicht Tabelle 5). Wir beginnen mit einem Vergleich zwischen der FM2-GCN-Architektur und der AGNN-Nachrichtenübermittlungsarchitektur. Wenn FM2-Modelle an sehr kleinen pbbg-AFs trainiert und getestet wurden (Ø $|A| = 25$), finden wir MCC-Werte von 0,54(C_1) und 0,558(D_1) für DS_{pr} und 0,57(C_1) für DC_{pr}. AGNN-Modelle hingegen zeigen dabei wesentlich höhere Werte von 0,998(C_1) und 0,962(D_1) für DS_{pr} und 0,997(C_1) für DC_{pr}. Für mittelgroße pbbg-AFs (Ø $|A| >= 200$) und DC_{pr} liegen die MCC-Werte für FM2-Modelle wesentlich niedriger zwischen 0,326(A_1) und 0,342 (A_4). AGNN-Modelle erreichen hier einen MCC-Wert von ungefähr 0,85 (C_2).

Wird ein Modell an einem pbbg-Datensatz trainiert und an einem iccma-Datensatz getestet, so sind dem Modell mehr als die Hälfte der AF-Typen grundsätzlich bekannt.[79] Allerdings scheitert die Anwendung des Lernen von Erdős-Rényi-AFs an sehr kleinen AFs auf die Einordnung von Argumenten aus größeren Erdős–Rényi-AFs, die Teil des Ausschnitts des ICCMA-2017-Datensatzes sind, den [KWT22] nutzen, mit einem MCC von nur -0,001(D_2); hingegen gelingt diese Anwendung bei Barabási-Albert-AFs mit 0,946(D_2).[80] Wird FM2 an kwt-Graphen mit je 151 Argumenten trainiert und getestet, erreicht es einen schlechten MCC-Wert von 0,364 (D_1), wohingegen ein AGNN sehr gute 0,927(D_1) erreicht.

Wird mit Testdaten getestet, die unbekannte AF-Typen enthalten, so bricht im Allgemeinen die Leistung beider Architekturen ein. Betrachten wir zunächst die Gesamtwerte für das Training mit pbbg- und das Testen mit iccma-AFs. Hier finden wir für FM2-Modelle MCC-Werte zwischen 0,146(A_5) und 0,211(D_1) und für AGNN-Modelle 0,507(D_1). Trainieren wir mit pbbg-AFs und testen wir an unbekannten kwt-AFs erreichen wir höhere MCC-Werte von 0,359(D_1) für FM2 und 0,631(D_1) für AGNN, was die beste Übertragungsleistung für FM2 darstellt. Drehen wir die Anordnung um, und trainieren mit kwt-AFs und testen an pbbg-AFs so zeigt der AGNN seine beste Übertragungsleistung überhaupt mit 0,693(D_1) und FM2 eine äußerst niedrige von 0,078(D_1).

Die MCC-Werte der tieferen res-GCNs (mit Ausnahme des balanzierten Testdatensatzes) müssen wir durch Unausgewogenheit der Testdatensätze bezüglich pa* mit einem Anteil von 3,6% an den Argumenten und besonders pa mit nur 1,2% achtsam mit anderen Werten vergleichen, wie ein Blick auf die Nebenkennzahlen TPR und TNR zeigt. Aber selbst unter dieser schlechten Rahmenbedingungen zeigen res-GCNs mit Werten zwischen 0,4(B_1) und 0,422(B_1) bei DC_{pr} moderate MCC-Werte beim Training mit und Testen an den großen ICCMA-2017-AFs. Diese MCC-Werte schlagen damit auch jene aus Fall A, die an wesentlich kleineren AFs getestet wurden und zeigen das Potential tieferer GCNs oder der DeepWalk-Graphenrohresiduen. Die höchste Leistung aller GCN-Architekturen setzt res-GCN4 mit 0,636(B_1)

[79] Wenngleich dies nicht unbedingt bedeutet, dass die Hälfte der Argumente in AFs mit bekannten AF-Typ sind.

[80] Im Abschnitt 4.1 untersuchen wir die Frage von distinkten Untergruppen innerhalb von AFs eines Typs genauer.

Experiment	Architektur	Trainingsset → Testset	MCC	Anmerkungen
A_1: DC_{pr}	FM1	$A_{L(pbbg)} \rightarrow A_{T(pbbg)}$	0,000	
	FM2	$A_{L(pbbg)} \rightarrow A_{T(pbbg)}$	0,326	TPR: 0,203; TNR: 0,980
A_4: DC_{pr}	FM2	$A_{L(pbbg)} \rightarrow A_{T(pbbg)}$	0,342	750 statt 500 Epochen
A_5: DC_{pr}	FM2	$A_{L1(half)} \rightarrow A_{T(iccma)}$	0,146	pbbg → iccma
		$A_{L(bal+)} \rightarrow A_{T(iccma)}$	0,156	pbbg → iccma
B_1: DC_{pr}	res-GCN4	$B_L \rightarrow B_T$	0,413	TPR: 0,693; TNR: 0,935
	res-GCN5	$B_L \rightarrow B_T$	0,422	TPR: 0,736; TNR: 0,930
	res-GCN6	$B_L \rightarrow B_T$	0,400	TPR: 0,718; TNR: 0,924
	res-GCN4	$B_{L(bal)} \rightarrow B_{T(bal)}$	0,636	TPR: 0,912; TNR: 0,710
B_2: DS_{pr}	res-GCN4	$B_L \rightarrow B_T$	0,1326	TPR: 0,240, TNR: 0,971
	res-GCN5	$B_L \rightarrow B_T$	0,1265	TPR: 0,229; TNR: 0,971
	res-GCN6	$B_L \rightarrow B_T$	0,1259	TPR: 0,227; TNR: 0,972
C_1: DC_{pr}	FM2	$C_L \rightarrow C_{T25}$	0,57	
	AGNN	$C_L \rightarrow C_{T25}$	0,998	
C_1: DS_{pr}	FM2	$C_L \rightarrow C_{T25}$	0,54	
	AGNN	$C_L \rightarrow C_{T25}$	0,997	
C_2: DC_{pr}	AGNN	$C_L \rightarrow C_{T25}$	0,99*	Daten aus Grafik
	AGNN	$C_L \rightarrow C_{T50}$	0,96*	Daten aus Grafik
	AGNN	$C_L \rightarrow C_{T100}$	0,85*	Daten aus Grafik
	AGNN	$C_L \rightarrow C_{T200}$	0,85*	Daten aus Grafik
D_1: DS_{pr}	FM2	$D_{L(pbbg)} \rightarrow D_{T(pbbg)}$	0,558	
		$D_{L(pbbg)} \rightarrow D_{T(kwt)}$	0,359	
		$D_{L(pbbg)} \rightarrow D_{T(iccma)}$	0,211	
		$D_{L(kwt)} \rightarrow D_{T(pbbg)}$	0,078	
		$D_{L(kwt)} \rightarrow D_{T(kwt)}$	0,364	
		$D_{L(kwt)} \rightarrow D_{T(iccma)}$	0,055	
	AGNN	$D_{L(pbbg)} \rightarrow D_{T(pbbg)}$	0,962	
		$D_{L(pbbg)} \rightarrow D_{T(kwt)}$	0,631	
		$D_{L(pbbg)} \rightarrow D_{T(iccma)}$	0,507	
		$D_{L(kwt)} \rightarrow D_{T(pbbg)}$	0,693	
		$D_{L(kwt)} \rightarrow D_{T(kwt)}$	0,927	
		$D_{L(kwt)} \rightarrow D_{T(iccma)}$	0,250	
D_2: DS_{pr}	AGNN	$D_{L(pbbg)} \rightarrow D_{T3(ABA2AF)}$	0,804	unbekannter Typ
		$D_{L(pbbg)} \rightarrow D_{T3(AdmBuster)}$	-0,110	unbekannter Typ
		$D_{L(pbbg)} \rightarrow D_{T3(BA)}$	0,946	bekannter Typ
		$D_{L(pbbg)} \rightarrow D_{T3(ER)}$	-0,001	bekannter Typ
		$D_{L(pbbg)} \rightarrow D_{T3(SCC)}$	0,135	bekannter Typ
		$D_{L(kwt)} \rightarrow D_{T3(ABA2AF)}$	0,680	unbekannter Typ
		$D_{L(kwt)} \rightarrow D_{T3(AdmBuster)}$	-0,454	unbekannter Typ
		$D_{L(kwt)} \rightarrow D_{T3(BA)}$	0,645	unbekannter Typ
		$D_{L(kwt)} \rightarrow D_{T3(ER)}$	0,009	unbekannter Typ

Tabelle 5: Zusammenfassung der Experimente der bisherigen Forschung.

beim Trainieren und Testen unter guten Bedingungen wegen der balanzierten Datensätze. Die Leistung der res-GCNs bei DS_{pr} ist mit MCC-Werten um 0,13 und TPR-Werten von rund 0,23-0,24 – wegen der erschwerten Umstände nicht überraschend – niedrig.

Für unsere Experimente ist zuallererst der direkte Bezug zu den Ergebnissen von `FM2` wichtig, da eines unserer Designs ihm nahezu entspricht (`GCN2LIN0`), sowie ein relativer Bezug zu den Ergebnissen der res-GCNs. Unsere drei Designs `GCN4LIN0`, `GCN5LIN0` und `GCN6LIN0` ähneln `res-GCN4`, `res-GCN5` und `res-GCN6` in Art und Tiefe der GNN-Schicht, ohne aber die Residuen-Technik zu nutzen, und stellen somit eine Zwischenform zwischen `FM2` und res-GCNs dar. Und mit `res-GCN4` wurde der bisherige MCC-Höchstwert für GCN-Architekturen erreicht. Gleichzeitig bilden die hervorragenden MCC-Werte der AGNNs Zielwerte, die wir zu erreichen und zu übertreffen suchen.

4 Analyse der Datensätze

Wir werden nun die von uns genutzten Datensätze in ihren Eigenschaften und in ihrer Zusammenstellung aus AF-Typen vorstellen. Über diese Eigenschaften werden wir dann versuchen lineare Zusammenhänge zwischen den Eigenschaften zu finden, die Einfluss auf die Lösung der AFs haben können. Zum Abschluss des Abschnitts erläutern wir unsere Methode die Auswahl von Abschlusstestdatensätzen repräsentativer zu machen.

4.1 Eigenschaften der Datensätze pbbg_2040, iccma_450 und kwt_2000

Um eine Vergleichbarkeit unserer Ergebnisse mit denen der bisherigen Forschung zu erreichen, arbeiten wir wie [KWT22] mit drei unterschiedlichen Datensätzen, die das Spektrum an AF-Typen abdecken und AFs verschiedener Komplexitätsgrade enthalten. Der erste – pbbg_2040 – enthält insgesamt 2040 AFs mit insgesamt 79.879 Argumenten und 420.083 Angriffen, die wir selbst generieren. Er ist zu gleichen Teilen durch die sechs Generatoren aus `AF-BenchGen` und `Probo` generiert worden. Die enthaltenen AFs sind aus drei Größenklassen: Pro AF-Typ haben je 200 AFs 50 Argumente, 100 AFs haben 25 Argumente und 40 AFs haben 5 bis 25 Argumente. Damit nimmt unser pbbg_2040-Datensatz Eigenschaften der pbbg-Datensätze der Fälle C und D auf, vermischt aber die Ansätze der Trainings- und Testdatensätze dieser Fälle und erhöht die durchschnittliche Argumentanzahl signifikant, so dass die AFs denen aus Fall A etwas ähnlicher werden. Der zweite – iccma_450[81] – besteht aus 450 AFs mit insgesamt 292.221 Argumenten und 23.730.588 Angriffen, die zu unterschiedlichen Teilen aus den elf ICCMA-Typen bestehen. Damit sind Direktvergleiche mit [KWT22] möglich und auch Vergleiche mit [MYNM20]. Der letzte –

[81] Es sind die dieselben AFs aus ICCMA 2017, die auch [KWT22] nutzen. Unter Fall D nennen wir ihn $D_{T(iccma)}$.

kwt_2000 – ist die Vereinigung der kwt-Trainings- und Test-AFs aus [KWT22] und besteht aus 302.000 Argumenten und 13.090.610 Angriffen. Mit diesen drei Datensätzen können wir Experimente produzieren, die mit den relevanten Experimenten der bisherigen vergleichbar sind. Der größte theoretische Unterschied ist, dass wir die Zugehörigkeit zur idealen Extension untersuchen. Für unsere Datensätze bedeutet allerdings pa fast immer ia, so dass wir unsere Ergebnisse zu SE_{id} direkt mit den Ergebnissen zu DS_{pr} der bisherigen Forschung und unter Vorbehalt auch mit denen zu DC_{pr} vergleichen können. Schauen wir uns die Eigenschaften der Datensätze in Tabelle 6 an, sehen wir, dass der iccma_450-Datensatz mit knapp unter 10% die anteilig wenigstens ia Argumente hat, der kwt_2000-Datensatz mit 45,8% die meisten, und der pbbg_2040-Datensatz liegt mit 22% dazwischen. Nicht in der Tabelle verzeichnet ist die Zahl der AFs, die außer der leeren Menge keine ideale Extension besitzen, deren Argumente alle ina sind. Dies sind 436 AFs aus pbbg_2040, 167 AFs aus iccma_450-AFs und 163 AFs aus dem kwt_2000-Datensatz. Diese AFs enthalten 22,5%, 28,1% und 8,2% der Argumente der jeweiligen Datensätze und ein naiver Nein-Entscheider löst sie ebenso gut wie ein korrekter Löser. Die Zahl der Argumente, die ina und pa sind, ist mit 0,2%, 0,3% und 0,1% der Argumente der jeweiligen Datensätze sehr niedrig. Für Argumente, die ina und pa* sind, liegen diese Anteile mit 10,6% für pbbg_2040, 16,5% für iccma_450 und 29,8% für kwt_2000 wesentlich höher. Die AFs aus pbbg_2040 bestehen aus 5 bis 51 Argumenten (Ø: 39,2; Median: 50). Die AFs in iccma_450 zeigen eine sehr große Varianz und sind zwischen 2 und 14.812 Argumenten groß (Ø: 649,4; Median: 321). Allein der größte Graph beinhaltet über 5% der Argumente des gesamten iccma_450-Datensatzes. In bestimmten Metriken wie ACC wird die korrekte Einordnung auf der Argumentebene ohne Gewichtung gemessen; dort entspricht der Einfluss eines Graphen seinem Anteil an den Gesamtargumenten. Dasselbe gilt für unsere im Allgemeinen ungewichtete Verlustfunktion während des Trainings des Modells. Alle AFs aus kwt_2000 bestehen aus 151 Argumenten. Der Median der Angriffe in den AFs in iccma_450 ist mit 3207 Angriffen pro AF wesentlich geringer als der Durchschnitt mit 52.735 Angriffen pro AF, pbbg_2040 zeigt einen wesentlich kleineren Unterschied von 111 zu 206 und kwt_2000 weist kaum einen Unterschied mit 6514 zu 6545 auf. Durchschnittlich gibt es im pbbg_2040-Datensatz knapp 5 Angriffe pro Argument, in iccma_450 knapp 81 und in kwt_2000 knapp 43. Ähnliches wie für die Angriffe gilt auch für den Durchmesser. Als längster der kürzesten Pfade zwischen zwei Argumenten gibt er an, wie viele Nachrichtenübermittlungsschritte nötig sind, damit die Einbettungen der am weitesten voneinander entfernten (aber über die gerichtete Angriffsrelation verbundenen) Argumente sich beeinflussen können. Nur für die AFs des iccma_450-Datensatzes weicht der Median mit 3 stark vom Durchschnitt mit 33,6 ab, für AFs des pbbg_2040-Datensatzes liegen beide bei ungefähr 6 und für kwt_2000-AFs liegt er bei ungefähr 3. Der Durchmesser der ungerichteten Transformation des AFs, die das KIPF-WELLING-GCN nutzt, ist genauso groß wie der Durchmesser des AGNNs, das durch seine Nachrichtenübermittlung auch in Gegenrichtung jeden Angriff implizit verdoppelt, der nicht vorher schon vorher reziprok war. Für einzel-

	Gesamt			Mediane				
Datensatz	#AFs	$\|A_{set}\|$	$\|R_{set}\|$	$\|A\|$	$\|R\|$	$\|R\|/\|A\|$	Durchm.	#ia
pbbg_2040	2040	79.879	420.083	50	111	2,9	6	8
iccma_450	450	292.221	23.730.588	321	3207	8	6	7,5
kwt_2000	2000	302.000	13.090.610	151	6514	43,1	3	75
	Anteil der Argumente in %			Arithmetische Mittel				
Datensatz	ia	pa	pa*	$\|A\|$	$\|R\|$	$\|R\|/\|A\|$	Durchm.	#ia
pbbg_2040	22,0	22,2	32,6	39,2	206	5,3	6,4	8,6
iccma_450	9,9	10,2	26,4	649,4	52.735	81,2	33,6	64,6
kwt_2000	45,8	45,9	75,0	151	6545	43,4	2,7	69,2

Tabelle 6: Überblick über die Eigenschaften der von uns genutzten Datensätze pbbg_2040, iccma_450 und kwt_2000. Alle Anteile sind in Prozent angegeben.

ne AFs kann der Durchmesser hier höher als im einfach gerichtetem Fall liegen, im Ganzen nimmt der Durchmesser aber ab. Für pbbg_2040 liegt der Median bei 4 und der Durchschnitt bei 4,8, für iccma_450 liegt der Median bei 4 und der Durchschnitt bei 33,8 und für kwt_2000 liegt der Median bei 2 und der Durchschnitt bei 2,4.

Wir fächern nun die Datensätze von pbbg_2040 und iccma_450 nach AF-Typ auf. Wir sehen für den pbbg_2040-Datensatz in Tabelle 7, dass die Erdős-Renyi- und SCC-AFs knapp zwei Drittel der Angriffe des Datensatzes aber nur ein Drittel der Argumente enthalten. Auch sind mit 7,8% und 6% nur wenige der Argumente ia. Dagegen enthalten die AFs vom Typ Barabási-Albert weniger als 5% der Angriffe des Datensatzes bei einem Sechstel der Argumente und mit 41,5% sind besonders viele dieser Argumente ia. Die Werte der AFs der anderen drei Typen liegen zwischen diesen Extremen. Die Grounded- und Stable-AFs ähneln etwas mehr dem Barabási-Albert-Typ und die Watts-Strogatz-AFs etwas mehr den Erdős-Renyi- und SCC-AFs. Eine hohe TPR ist also ohne die korrekte Einordnung der meisten ia Argumente in Barabási-Albert-AFs nicht möglich. Betrachten wir den Durchmesser, so liegt der Median für Erdős-Renyi- und SCC-AFs bei 3 und für die anderen Typen bei 7 und 8.

Der iccma_450-Datensatz in Tabelle 8 ist sehr inhomogen und zeigt eine große Varianz zwischen den Typen als auch innerhalb der Typen. Die Anzahl an AFs pro Typ variiert von 5 für AdmBuster bis zu 68 für Watts-Strogatz. Der Median der Anzahl an Argumenten pro Typ variiert von 100 für Traffic-AFs bis zu 4000 für AdmBuster-AFs und die Größe selbst von 2 bis zu 14.812 Argumenten (letztere AFs sind beide vom Traffic-Typ). Die AFs vom ABA2AF-Typ stellen nur 4,4% der Argumente des Datensatzes, aber fast ein Drittel aller Angriffe. Wohingegen Planning2AF- und Traffic-AFs jeweils rund 10% der Argumente des Datensatzes beinhalten, aber nur 0,2% der Angriffe. Die 62 Barabási-Albert-AFs bestehen zu 43,3% aus ia Argumenten, stellen insgesamt nur 2,7% der Argumente des Datensatzes. Zusammengenommen sind das 11,8% der ia Argumente des Datensatzes. Die 5 AdmBuster-AFs bestehen sogar zur Hälfte aus ia Argumenten, stellen 7,9% der Argumente des Da-

	Gesamt			Mediane				
AF-Typ	#AFs	$\|A_{typ}\|$	$\|R_{typ}\|$	$\|A\|$	$\|R\|$	$\|R\|/\|A\|$	Durchm.	#ia
Barabási-Albert	340	13.311	19.045	50	54	1,5	7	15
Erdős-Rényi	340	13.348	155.782	50	311	9,4	3	0
Grounded	340	13.264	41.854	50	136	2,8	8	12
SCC	340	13.343	130.591	50	292	7,1	3	2
Stable	340	13.302	31.177	50	83	2,0	7	10
Watts-Strogatz	340	13.311	41.634	50	100	3,0	8	7

	Anteil der Argumente			Arithmetische Mittel				
AF-Typ	ia	pa	pa*	$\|A\|$	$\|R\|$	$\|R\|/\|A\|$	Durchm.	#ia
Barabási-Albert	41,5	41,5	82,1	39	56	1,4	6,7	16,3
Erdős-Rényi	7,8	7,9	9,0	39	458	11,7	3,9	3,1
Grounded	31,3	31,3	31,7	39	123	3,2	7,9	12,2
SCC	6,0	6,3	9,5	39	384	9,8	3,7	2,4
Stable	27,8	28,7	39,4	39	92	2,3	7,6	10,9
Watts-Strogatz	17,5	17,8	24,2	39	122	3,1	8,6	6,8

	Anteil am Datensatz			
AF-Typ	AFs	|A|	|R|	ia
Barabási-Albert	16,7	16,7	4,5	31,5
Erdős-Rényi	16,7	16,7	37,1	5,9
Grounded	16,7	16,6	10,0	23,7
SCC	16,7	16,7	31,1	4,6
Stable	16,7	16,7	7,4	21,1
Watts-Strogatz	16,7	16,7	9,9	13,3

Tabelle 7: Eigenschaften der AFs der in pbbg_2040 genutzen AF-Typen. Alle Anteile sind in Prozent angegeben.

tensatzes, und beinhalten so 39,6% der ia Argumente. Die Erdős-Renyi-AFs stellen hingegen ähnliche 7,6% der Argumente des Datensatzes, aber sie bestehen nur zu 0,1% aus ia Argumenten. Ihr Anteil an den ia Argumenten insgesamt ist mit 0,1% vernachlässigbar. ABA2AF-, SCC-, SemBuster-, und WattsStrogatz-AFs haben ähnlich wenig ia Argumente.[82] Die anderen fünf Typen liegen bezüglich ihrer Anteile zwischen den Extremen. Diese ungleiche Verteilung der ia Argumente auf die AF-Typen hat Auswirkungen auf die möglichen Leistungen eines trainierten Modells einer bestimmten Architektur. Falls z. B. ein Modell stest die ia Argumente aller AdmBuster-AFs falsch einordnet, kann es hier keinen hohen MCC-Wert oder TPR-Wert mehr erreichen, da mindestens 39,6% von P dann stets FN sind. Falls das Modell sich zu einem Nein-Entscheider entwickelt, sinkt der ACC-Wert gegen 0,5 und der Verlust während des Trainings steigt. Für Erdős-Rényi-, SCC- oder Watts-Strogatz-AFs und selbstverständlich auch für SemBuster-AFs gilt dies nicht, da bei ihnen Nein-Entscheider hohe ACC-Werte und niedrigen Verlust erreichen. Wir werden später auch die Kennzahlen nach Typen auffächern und dann kann die hohe Unausgewogenheit die üblichen Auffälligkeiten bei den Kennzahlen auslösen.

Die Abweichung des Median des Durchmessers von AFs des iccma_450-Datensatzes ist zum großen Teil auf die AFs vom AdmBuster-Typ zurückzuführen, deren Durchmesser im Median 2000 beträgt. Ihre spezifische Struktur und großer Durchmesser macht eine approximative Lösung über Nachbarschaftsagglomeration besonders schwierig. Grounded-, Planning2AF- und Traffic-AFs haben auch recht hohe Durchmessermediane von 14 bis 16, Barabási-Albert-AFs liegen mit 9 in der Mitte und die anderen Typen haben Mediane von 2 bis 6. Wenn wir dieselben Typen aus dem pbbg_2040-Datensatz und dem iccma_450-Datensatz vergleichen, so fällt auf, dass sich die zu $|A|$ relativen Eigenschaften %ia und den $|R|/|A|$-Median der gleichen Typen stark voneinander unterscheiden. Nur bei Barabási-Albert-AFs gibt es mit 41,5 %ia im pbbg_2040-Datensatz zu 43,3 %ia im iccma_450-Datensatz und 1,4 $|R|/|A|$-Median zu 1,5 $|R|/|A|$-Median keinen großen Unterschied. Bei Grounded-AFs fällt %ia schon von 31,3 auf 5,8 und der $|R|/|A|$-Median steigt um den Faktor 20 von 2,8 auf 59,5. Ähnliches finden wir bei den Stable-AFs. Hier fällt %ia von 27,8 auf 6,7 und der $|R|/|A|$-Median steigt weniger stark von 2,0 auf 8,1. Der Anteil an ia Argumenten fällt in Watts-Strogatz-AFs beim Vergleich zwischen dem pbbg_2040-Datensatz und dem iccma_450-Datensatz stark um den Faktor 35 von 17,5% auf 0,5%, der $|R|/|A|$-Median steigt weniger stark von 3,0 auf 11,0. Am stärksten unterscheiden sich Erdős-Rényi- und SCC-AFs in den beiden Datensätzen. In Erdős-Rényi-AFs bricht %ia um den Faktor 78 von 7,8% auf 0,1% ein und der $|R|/|A|$-Median steigt von 9,4 auf 99,7. Ähnlich fällt in SCC-AFs %ia von 6,0% auf 0,1% ein und der $|R|/|A|$-Median steigt hier von 7,1 auf 61,6. Unser pbbg_2040-Datensatz hat einen geringeren Anteil an pa Argumenten als der in [KWT22] auch für Übertragungsexperimente genutzte $D_{L(pbbg)}$. Damit gehen wir davon aus, dass auch die

[82] Diese hohe Unausgeglichenheit der Klassen hat Auswirkungen auf die MCC-Werte und ihre Vergleichbarkeit untereinander, wenn wir in unseren Experimenten die MCC-Median-Werte nach AF-Typen auffächern.

	Gesamt			Mediane				
AF-Typ	#AFs	$\|A_{typ}\|$	$\|R_{typ}\|$	$\|A\|$	$\|R\|$	$\|R\|/\|A\|$	Durchm.	#ia
ABA2AF	34	12.772	7.732.612	450	192.669	429,0	2	10
AdmBuster	5	23.000	34.490	4000	5998	1,5	2000	2000
Barabási-Albert	62	7942	11.549	121	191	1,4	9	53
Erdős-Rényi	64	22.073	2.733.626	352	39.917	99,7	2	0
Grounded	15	50.561	3.007.317	1948	71.714	35,8	16	147
Planning2AF	53	27.698	52.484	265	461	1,8	14	40
SCC	34	53.886	7.077.112	1128	73.428	61,6	4	0
SemBuster	10	11.910	2.564.810	1050	126.050	117,7	3	0
Stable	59	27.245	211.313	437	3550	8,1	6	10
Traffic	46	31.134	52.675	100	189	1,7	15	16,5
Watts-Strogatz	68	24.000	252.600	400	3600	11,0	5	0

	Anteil der Argumente			Arithmetische Mittel				
AF-Typ	ia	pa	pa*	$\|A\|$	$\|R\|$	$\|R\|/\|A\|$	Durchm.	#ia
ABA2AF	3,2	3,2	3,4	376	227.430	605,4	2,4	12,1
AdmBuster	50,0	50,0	50,0	4600	6898	1,5	2300	2300
Barabási-Albert	43,3	43,3	83,0	128	186	1,5	9,2	55,4
Erdős-Rényi	0,1	0,1	0,1	345	42.713	123,8	2,3	0,3
Grounded	5,8	5,8	5,8	3371	200.488	59,5	17,3	194,3
Planning2AF	13,4	13,5	64,9	523	990	1,9	14,9	70,1
SCC	0,1	0,2	0,4	1585	208.150	131,3	4,4	2,1
SemBuster	0,0	0,0	66,7	1191	256.481	215,3	3,0	0,0
Stable	6,7	9,5	23,1	462	3582	7,8	7,1	31,1
Traffic	16,2	16,3	74,5	677	1145	1,7	18,0	109,6
Watts-Strogatz	0,5	0,6	0,9	353	3715	10,5	5,5	1,7

	Anteil am Datensatz			
AF-Typ	AFs	|A|	|R|	ia
ABA2AF	7,6	4,4	32,6	1,4
AdmBuster	1,1	7,9	0,1	39,6
Barabási-Albert	13,8	2,7	0,0	11,8
Erdős-Rényi	14,2	7,6	11,5	0,1
Grounded	3,3	17,3	12,7	10,0
Planning2AF	11,8	9,5	0,2	12,8
SCC	7,6	18,4	29,8	0,3
SemBuster	2,2	4,1	10,8	0,0
Stable	13,1	9,3	0,9	6,3
Traffic	10,2	10,7	0,2	17,3
Watts-Strogatz	15,1	8,2	1,1	0,4

Tabelle 8: Eigenschaften der AFs der in iccma_450 genutzen AF-Typen. Alle Anteile sind in Prozent angegeben

Erdős-Rényi-AFs in diesem Datensatz eher denen aus pbbg_2040-Datensatz ähneln als denen aus iccma_450-Datensatz. Wir vermuten, dass z. B. der in $D_2(L(\text{pbbg}) \rightarrow T(\text{Barabási-Albert}))$ gefundene hohe MCC-Wert von 0,946 auf dieser relativen Eigenschaften*stabilität* in Bezug auf Skalierung der Anzahl der Argumente der Barabási-Albert-AFs beruht. Hingegen unterscheiden sich z. B. SCC-AFs aus unserem eigenen pbbg_2040-Datensatz stärker von SCC-AFs aus dem iccma_450-Datensatz als zwei unterschiedliche AF-Typen im pbbg_2040-Datensatz. Wir vermuten deshalb weiter, dass der niedrige MCC-Wert von 0,135 in $D_2(L(pbbg) \rightarrow T(SCC))$ auf dieser hohen Eigenschaften*instabilität* von SCC-AFs in Bezug auf Skalierung der Anzahl der Argumente fußt. Die AGGN-Modelle erkannten vermutlich im letzten Teilexperiment die Test-AFs als SCC-AFs wieder. In der Beprechung der Experimente E werden wir unter anderem versuchen, überraschende Testergebnisse auf die Eigenschaften der Datensätze und ihrer AF-Typen-Untermengen zurückzuführen.

4.2 Die Suche nach linearen Zusammenhängen

Eine Möglichkeit, um lineare Zusammenhänge einzelnen Eigenschaften besser zu verstehen, sind Korrelationskoeeffizienten. Über sie können wir die linearen Zusammenhänge aufdecken und charakteristische Beziehungsmuster für AF-Typen herausarbeiten. Nicht-lineare Zusammenhänge bleiben dabei selbstverständlich verdeckt. Wir fassen die Korrelationskoeffizienten jeweils zu Korrelationsmatrizen zusammen. Falls zwei Eigenschaften in einer Matrix perfekt (nahezu perfekt) miteinander korreliert sind, verhalten sie sich in den anderen Beziehungen dieser Matrix gleich (sehr ähnlich) und wir können oft die Beziehungen der einen Eigenschaft unter denen der anderen subsumieren. So haben in den meisten der folgenden Matrizen der Anteil an ia Argumenten pro AF (kurz %ia) und der Anteil an pa Argumenten pro AF (kurz %pa) mit 0,99 bis 1,0 einen fast perfekten[83] Korrelationskoeffizienten und wir sprechen %pa nur in den Einzelfällen an, wenn der Koeffizient davon abweicht. Weiterhin ist die durchschnittliche Anzahl an Angriffen pro Argument $|R|/|A|$ direkt von der absoluten Anzahl an Angriffen des AFs $|R|$ und der absoluten Anzahl an Argumenten des AF $|A|$ abhängig. Falls z. B. $|A|$ stets gleich bleibt (wie bei allen kwt-AFs), dann verhält sich $|R|/|A|$ zu anderen Eigenschaften genauso wie $|R|$. Als Signifikanzschwelle nutzen wir 3σ und lassen Einträge mit niedrigerer Signifikanz leer. Wir erwarten, dass wir zumindest für die gemischten Datensätze pbbg_2040 und iccma_450 eine Bandbreite an unterschiedlichen Werten für Beziehungen von %ia/%pa zu den anderen Eigenschaften finden werden. Falls wir wider Erwarten doch eine solche sehr starke Korrelation finden, könnte sie bei der Einordnung von ia Argumenten helfen. Wir könnten sie dann in einem Folgeprojekt entweder als Zusatzinformation für die Eingangsmerkmale nutzen oder sogar einen naiven Argument-Klassifizierer augmentieren, wie ihn [KWT22] mit der

[83] Wir bezeichnen Werte von $|1,0|$ als perfekte Korrelation, $|0,99|$ als fast perfekt, $|0,8| - |0,98|$ als sehr stark, $|0,6| - |0,79|$ als stark, $|0,4| - |0,59|$ als moderat, $|0,2| - |0,49|$ als schwach, $|0,01| - |0,19|$ als sehr schwach und $|0,0|$ als unkorreliert. Das Vorzeichen bestimmt, ob es eine positive oder negative Korrelation ist.

Information des durchschnittlichen Ein- und Ausgangsgrad von pa Argumenten im Datensatz nutzten.

Wir erstellen Korrelationsmatrizen sowohl für alle Datensätze gemeinsam, jeden Datensatz einzeln und für jeden AF-Typ der pbbg_2040- und iccma_450-Datensätze einzeln. Bei der Besprechung der Befunde beschränken wir uns auf solche von besonderem Interesse. Dies sind unüblich hohe oder niedrige Werte, sowie die Beziehungen von %ia zu den anderen Eigenschaften. Fassen wir zunächst alle drei Datensätze (siehe erste Matrix der Abb. 2 auf Seite 45) zu einem zusammen, sind nur %ia und %pa perfekt miteinander korreliert. Die Beziehung zwischen %ia zu %pa* ist mit 0,68 eine starke positive Korrelation. Nur wenige andere Beziehungen sind moderat oder stärker positiv korreliert: $|A|$ und $|R|$ mit 0,57, $|R|$ und $|R|/|A|$ mit 0,62. $|A|$ und der Durchmesser sind mit 0,39 nur schwach positiv korreliert. Interessant ist, dass über den ganzen Datensatz kein linearer Zusammenhang zwischen %ia (und damit %pa) oder %pa* zu den Eigenschaften $|A|$, $|R|$, $|R|/|A|$ oder Durchmesser zu finden ist. Dies ist durch die Beziehungsmuster der AF-Typen verursacht, in denen diese Eigenschaften ganz unterschiedlich und auch gegensätzlich miteinander korreliert sein können.

Vergleichen wir die drei Datensätze miteinander (siehe die anderen drei Matrizen der Abb. 2 auf Seite 45), so fallen Unterschiede in den Beziehungen zwischen $|R|$ und $|R|/|A|$ auf. Im kwt_2000-Datensatz sind sie wegen $|A| = 151$ perfekt miteinander korreliert. Im pbbg_2040-Datensatz korrelieren sie mit 0,98 sehr stark und dementsprechend ähneln sich beide Eigenschaften in den Beziehungen zu den anderen Eigenschaften stark (außer zu $|A|$). Dagegen sind sie im iccma_450-Datensatz mit 0,63 nur stark positiv miteiander korrelliert. Größere Unterschiede finden wir in den Beziehungen des Durchmessers. Zwischen ihm und $|R|$ ($|R|/|A|$) finden wir im iccma_450-Datensatz keinen signifikanten linearen Zusammenhang, im pbbg_2040-Datensatz ist er mit -0,43 (-0,5) moderat negativ und im kwt_2000-Datensatz ist er moderat positiv mit 0,66 (0,66). Der Unterschied zwischen den Extremen ist mit 1,09 hoch. Die drei Datensätze verhalten sich hier also ganz unterschiedlich, obwohl viele Graphen im iccma_450-Datensatz vom selben AF-Typ wie die im pbbg_2040-Datensatz sind. Der Durchmesser und %ia sind im pbbg_2040-Datensatz mit 0,24 schwach positiv korreliert, im iccma_450-Datensatz mit 0,18 nur sehr schwach positiv und im kwt_2000-Datensatz mit -0,32 schwach negativ. %ia ist mit $|A|$ nur im pbbg_2040-Datensatz mit 0,21 knapp schwach negativ korreliert und in den anderen beiden Datensätzen ohne signifikante Korrelation. Mit $|R|$ und $|R|/|A|$ ist %ia im pbbg_2040-Datensatz und kwt_2000-Datensatz moderat negativ korreliert, im iccma_450-Datensatz hingegen nur sehr schwach und schwach negativ, die drei Datensätze verhalten sich in diesem Fall wieder ähnlicher. Mit der Anzahl der Angriffe sinkt in allen Datensätzen der Anteil an ia Argumenten und umgekehrt, allerdings im pbbg_2040-Datensatz und kwt_2000-Datensatz stärker als im iccma_450-Datensatz. Als letzte Eigenschaft ist %ia mit %pa* im pbbg_2040-Datensatz mit 0,62 stark positiv korreliert, ähnlich im iccma_450-Datensatz mit 0,56, aber im kwt_2000-Datensatz mit 0,31 nur schwach positiv. Letzteres ist auffällig, da jedes ia Argument

auch pa* ist.

Wir fächern den pbbg_2040-Datensatz nun mit sechs Korrelationsmatrizen nach AF-Typen auf (vgl. Abb. 3 auf Seite 46). Für alle Typen sind %ia und %pa perfekt oder mit 0,99 fast perfekt korreliert. AFs vom Barabási-Albert-Typ zeigen mit 0,51 eine nur moderate Korrelation zwischen $|R|$ und $|R|/|A|$, bei den anderen ist sie mit Werten von mindestens 0,9 sehr stark. In allen AF-Typen außer dem Barabási-Albert-Typ verhalten sich dementsprechend $|R|$ und $|R|/|A|$ meist ähnlich. So ist die Korrelation zwischen diesen zwei Eigenschaften und %ia jeweils moderat bis stark negativ mit Werten zwischen -0,78 und -0,44, wobei die beiden Koeffizienten für jeden AF-Typ stets nahe (<0,1) beieinander liegen. Dies ist anders für den Barabási-Albert-Typ. Hier sind $|R|$ und %ia mit -0,29 schwach negativ und $|R|/|A|$ und %ia mit -0,79 stark negativ korreliert mit einem wesentlich größeren Unterschied von 0,3. Für %ia und den Durchmesser gleichen sich die Korrelationskoeffizienten der Barabási-Albert-, Grounded- und Stable-AFs mit schwach negativen -0,38, -0,36 und -0,36. Die der anderen drei sind sind hingegen schwach, moderat und stark positiv mit 0,18 für den Watts-Strogatz-Typ, 0,31 für den SCC-Typ und 0,6 für den Erdős-Renyi-Typ. Während für den einen AF-Typ also der Anteil an ia Argumenten mit steigendem Durchmesser abnimmt, zeigen AFs eines anderen Typs das gegenteilige Verhalten. Der Unterschied zwischen dem Minimum von -0,38 und dem Maximum von 0,6 ist mit 0,98 groß. Noch größer ist er nur in der Beziehung zwischen %ia und %pa*. In Erdős-Renyi- und Grounded-AFs ist Korrelation mit jeweils 0,91 sehr stark positiv, für SCC-AFs ist sie mit 0,75 noch stark positiv, für Watts-Strogatz- und Stable-AFs ist sie mit 0,44 und 0,33 nur noch moderat positiv. Es fallen aber erneut die Barabási-Albert-AFs mit einer sehr stark negativen Korrelation von -0,81 auf. Üblicherweise sind %ia und %pa* positiv korreliert, da alle ia Argumente auch pa* sind. Hier ist es aber der Fall, dass sich bei bei niedrigen %ia-Werten die bevorzugten Extensionen stärker voneinander unterscheiden, da es mehr Argumente gibt, die nicht Teil aller Extensionen sind, und es bei höheren %ia-Werten weniger Argumente gibt, für die das gilt. Insgesamt gibt es für die Beziehung einen sehr großen Unterschied von 1,72 zwischen dem Minimum von -0,81 und dem Maximum von 0,91. Wie aus der negativen Korrelation zwischen %ia und %pa* für Barabási-Albert-AFs zu erwarten ist, unterscheiden sich %ia und %pa* auch stark in ihren Beziehungen zu den anderen Eigenschaften. So sind %pa* und $|R|/|A|$ mit 0,87 sehr stark positiv korreliert statt mit -0,79 stark negativ, und %pa* und der Durchmesser sind mit 0,42 moderat positiv korreliert statt mit -0,38 schwach negativ.

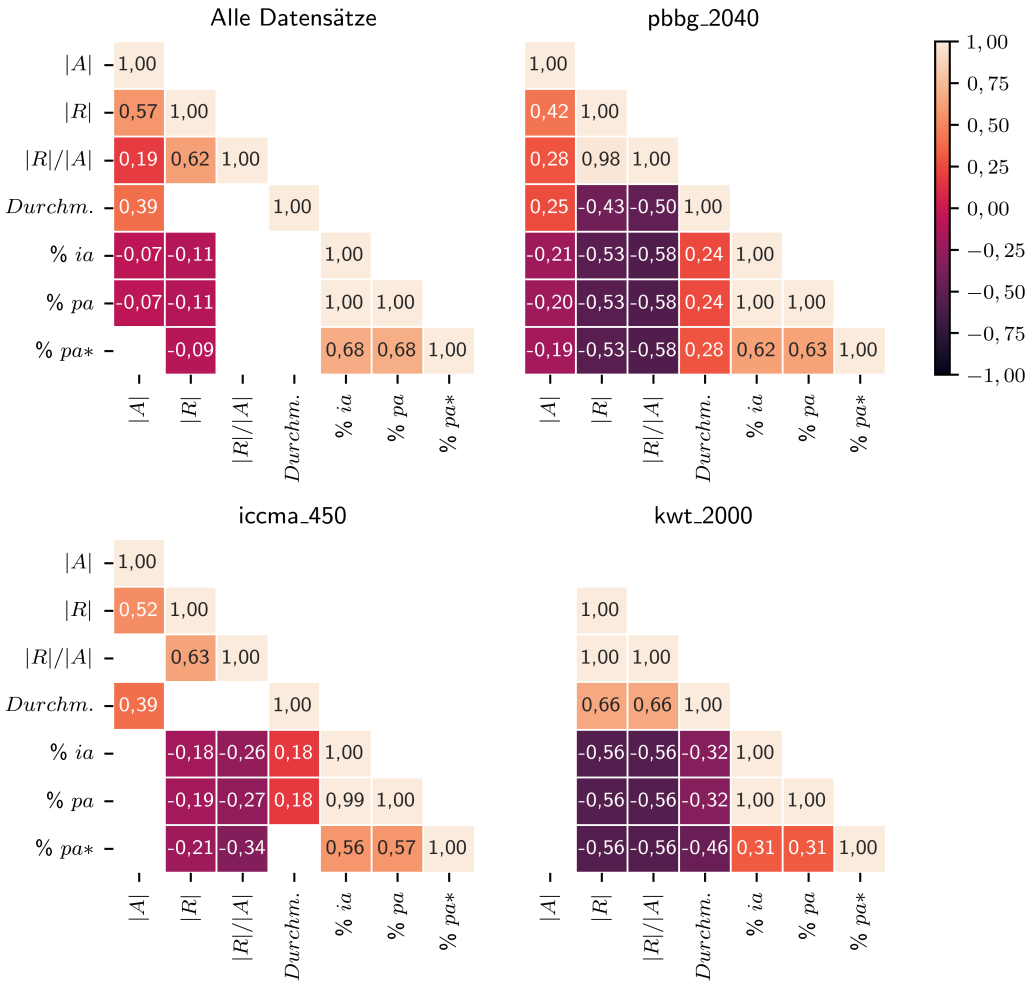

Abbildung 2: Korrelationsmatrizen der Datensätze pbbg_2040, iccma_450 und kwt_2000. Zunächst für alle Datensätze vereint, dann jeweils einzeln. Nicht signifikante Einträge wurden ausgelassen.

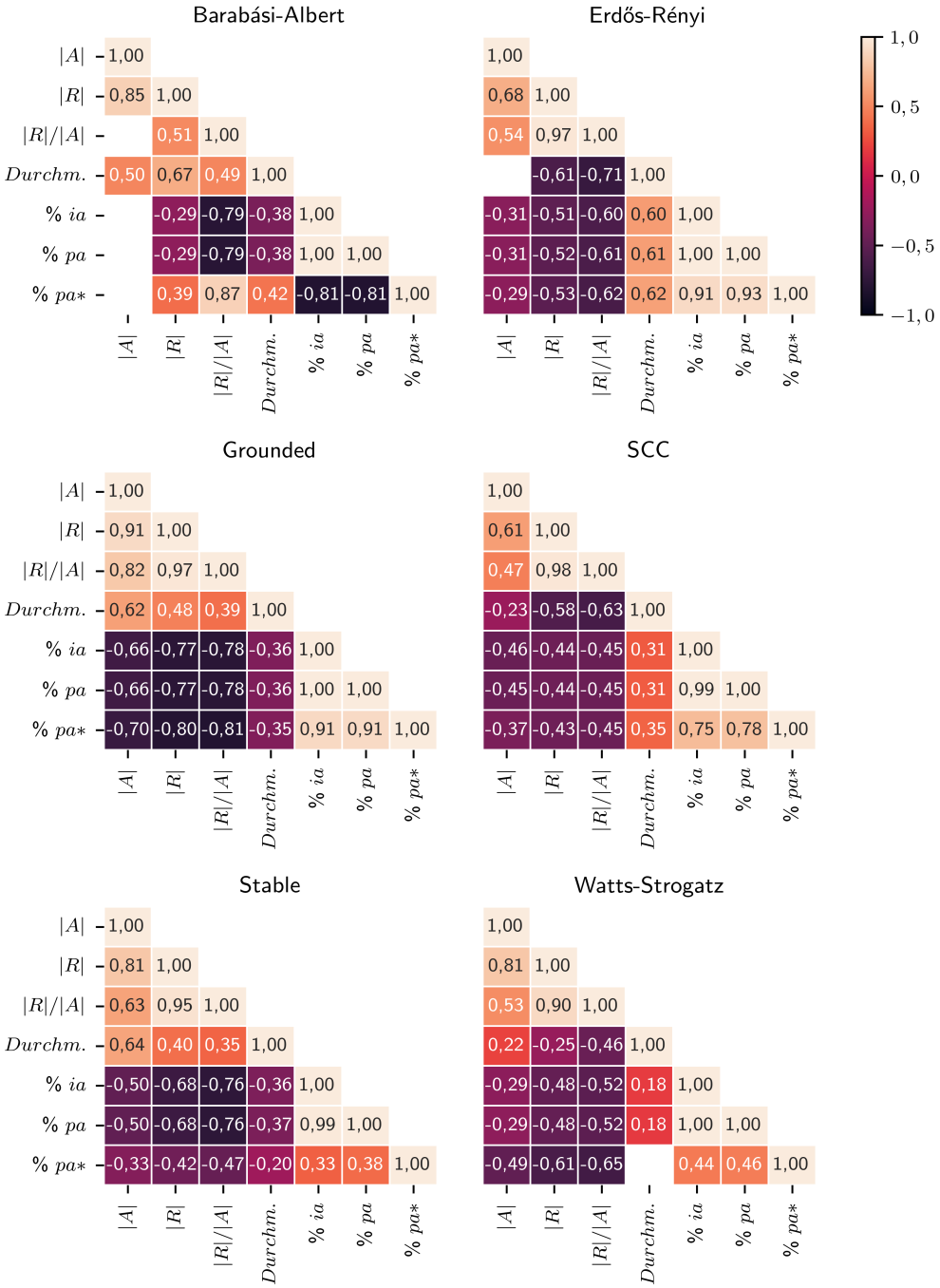

Abbildung 3: Korrelationsmatrizen der sechs AF-Typen des pbbg_2040-Datensatzes. Nicht signifikante Einträge wurden ausgelassen.

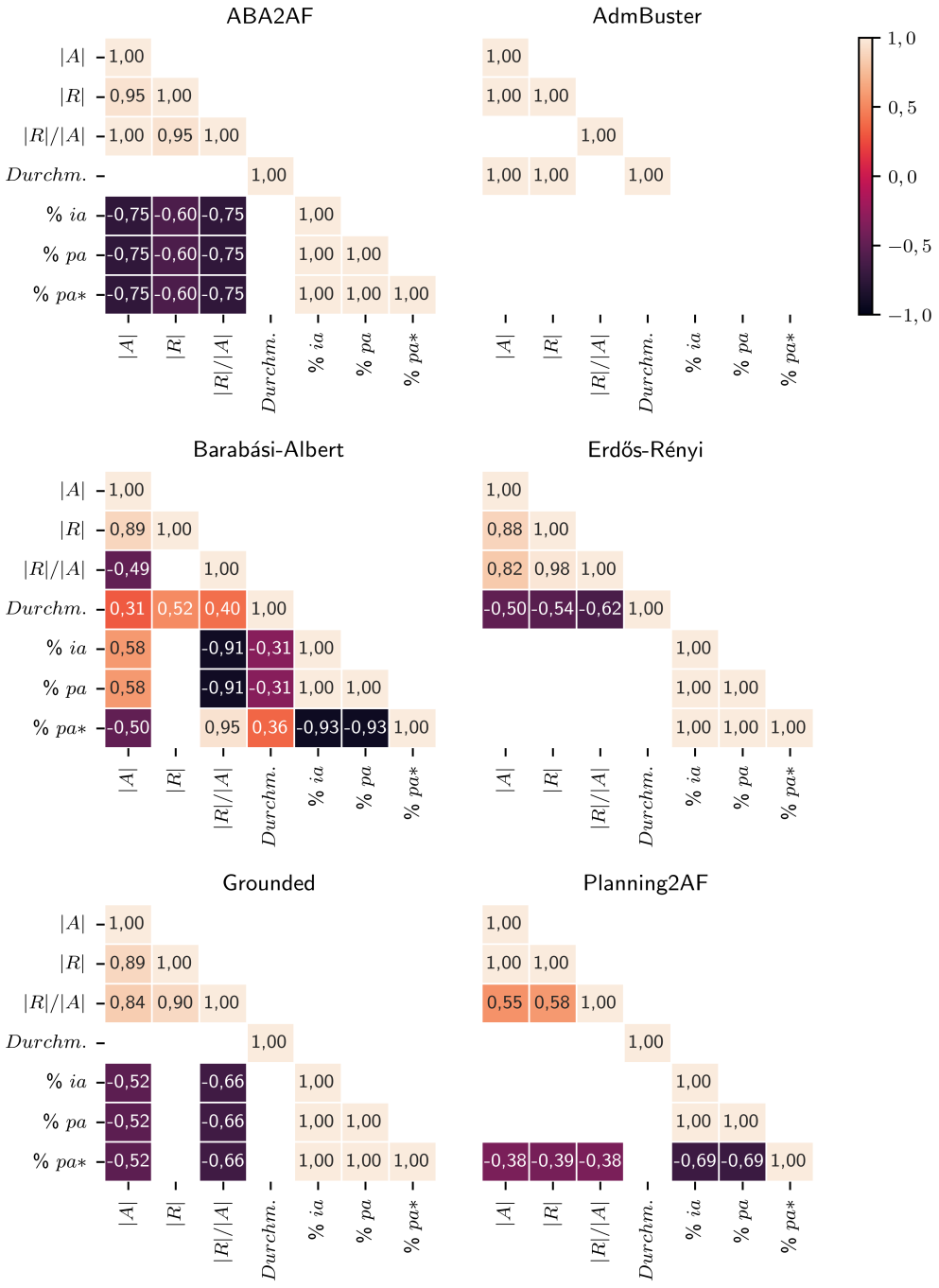

Abbildung 4: Korrelationsmatrizen der ersten Hälfte der elf AF-Typen des iccma_450-Datensatzes. Nicht signifikante Einträge wurden ausgelassen.

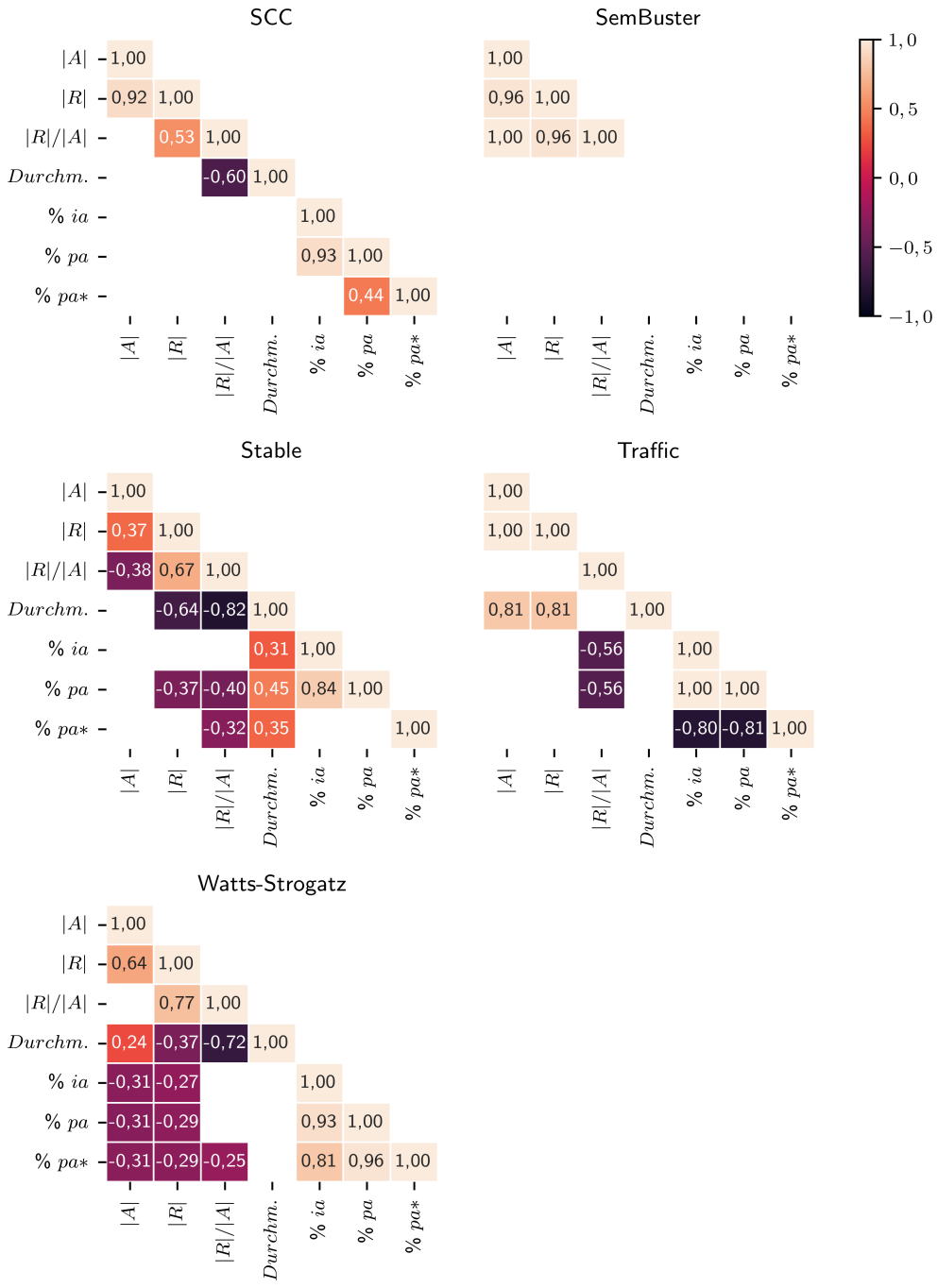

Abbildung 5: Korrelationsmatrizen der zweiten Hälfte der elf AF-Typen des iccma_450-Datensatzes. Nicht signifikante Einträge wurden ausgelassen.

Für den iccma_450-Datensatz (siehe Matrizen in den Abb. 4 auf Seite 47 und 5 auf Seite 48) fällt zuerst auf, dass viele Einträge leer bleiben. Dies hat zwei Ursachen. Zum einen fehlen bei den Matrizen für AdmBuster und SemBuster Koeffizienten für die Beziehungen von %ia, %pa und %pa* zu anderen Eigenschaften, da erstere wegen der spezifischen Struktur der AFs konstant sind. Gleiches gilt bei SemBuster-AFs auch für den Durchmesser. Die anderen leeren Stellen sind unserer Signifikanzschwelle und der geringen Anzahl an AFs pro Typ geschuldet.

Betrachten wir die Korrelationsmatrizen der mit den sechs Generatoren aus `Probo` und `AFBenchGen` erstellten AF-Typen, ähneln sie grob den Matrizen derselben AF-Typen aus dem pbbg_2040-Datensatz mit gewissen Unterschieden. So zeigt der Barabási-Albert-Typ mit -0,93 eine sehr stark negative Korrelation zwischen %ia und %pa*, eine ähnlich negative mit -0,91 zwischen %ia und $|R|/|A|$ und mit 0,95 eine stark positive zwischen %pa* und $|R|/|A|$. Anstatt leerer Einträge in Bezug von $|A|$ auf andere Eigenschaften haben wir aber hier leere Einträge in Bezug von $|R|$ zu anderen Eigenschaften. Bei Erdős-Renyi-AFs sind die Anteile %ia, %pa und %pa* miteinander perfekt korreliert, da aber nur ein einziges AF Argumente enthält die ia sind, und kein weiteres Argument der AFs dieses Typs pa oder pa* sind, existieren keine anderen signifikanten Korrelationen, und deren Einträge sind leer. Wie im pbbg_2040-Datensatz haben wir auch hier negative Korrelation zwischen dem Durchmesser und den anderen Eigenschaften, wobei sie hier moderat statt stark negativ sind. Auch bei den Grounded-AFs sind alle pa* Argumente ebenfalls ia. Außer der sehr starken positiven Korrelation zwischen $|R|$ und $|R|/|A|$, konnten sowohl für die Beziehungen von $|R|$ zu anderen Eigenschaften als auch des Durchmessers zu anderen Eigenschaften keine signifikanten Ergebnisse berechnet werden. %ia korreliert hier ähnlich wenn auch abgeschwächt wie die AFs des AF-Typ-Pendants im pbbg_2040-Datensatz moderat negativ und stark negativ mit $|A|$ und $|R|/|A|$. Für den SCC-Typ sind die meisten Einträge leer. Der lineare Zusammenhang zwischen %ia und %pa ist mit 0,93 etwas schwächer als im pbbg_2040-Datensatz, ansonsten zeigt keine Beziehung von %ia zu einer anderen Eigenschaften einen signifikanten Korrelationskoeffizienten. Dafür zeigen der Durchmesser und $|R|/|A|$ eine sehr ähnliche starke negative Korrelation mit -0,6 wie im pbbg_2040-Datensatz. Verschieden sind die Relationen zwischen $|A|$, $|R|$ und $|R|/|A|$. $|A|$ und $|R|$ haben einen Korrelationskoeffizienten von 0,92 statt 0,61 und umgekehrt haben $|R|$ und $|R|/|A|$ nun einen von 0,53 statt 0,98, was eventuell auf der am Ende von Abschnitt 4.1 erwähnten Eigenschafteninstabilität beruhen könnte. Für Watts-Strogatz-AFs ist der Korrelationskoeffizient von %ia und %pa mit 0,93 im iccma_450-Datensatz gegenüber 1,00 im pbbg_2040-Datensatz geringer, dafür ist der von %ia und %pa* mit 0,81 gegenüber 0,44 wesentlich höher. $|R|$ zu $|R|/|A|$ ist hier mit 0,77 statt 0,9 nur stark positiv statt sehr stark positiv korreliert. Der Durchmesser und $|R|/|A|$ sind im iccma_450-Datensatz mit -0,72 statt mit -0,46 wie im pbbg_2040-Datensatz stark negativ statt moderat negativ korreliert. Wir finden also trotz der allgemeinen Ähnlichkeiten deutliche Unterschiede zwischen den Korrelationsmatrizen desselben AF-Typs.

Für unseren pbbg_2040-Datensatz können wir davon ausgehen, dass sich die Er-

gebnisse bezüglich der idealen Extension auf die skeptische Akzeptanz bezüglich der bevorzugten Extension übertragen lassen, da der Korrelationskoeffizient zwischen %ia und %pa für alle AF-Typen perfekt oder fast perfekt ist und weil nur 99% der pa Argumente auch ia sind.[84] Dies ist im iccma_450-Datensatz nicht der Fall für SCC-, Stable- und Watts-Strogatz-AFs. Sie zeigen zum einen nur sehr starke Korrelationen mit 0,93, 0,84 und 0,93 und zum anderen sind nur 88%, 71% und 75% der pa Argumente auch ia.[85] Wir können für den iccma_450-Datensatz deshalb nur von einer eingeschränkten Übertragbarkeit unserer Ergebnisse bezüglich der idealen Extension auf die skeptische Akzeptanz bezüglich der bevorzugten Extension schließen. Schließen wir diese AF-Typen aber aus, erwarten wir, dass ia und pa aus praktischer Sicht gleichbedeutend sind und unsere Ergebnisse auch auf DS_{pr} übertragbar sind.

Schauen wir uns die drei verbleibenden AF-Typen ABA2AF, Planning2AF und Traffic im Detail an. Für alle drei gibt es eine perfekte Korrelation zwischen %ia und %pa. Für ABA2AF-AFs kommt hier noch eine perfekte Korrelation zwischen %ia und %pa* hinzu. Planning2AF-AFs zeigen mit -0,69 hingegen eine stark negative Korrelation zwischen %ia und %pa*, Traffic-AFs zeigen mit -0,8 sogar knapp sehr stark negative Korrelationen. $|R|/|A|$ ist für ABA2AF-AFs perfekt mit $|A|$ und sehr stark positiv mit $|R|$ korreliert. SemBuster- und AdmBuster-AFs mit ihrer Regelmäßigkeit gleichen in diesen Korrelationen ABA2AF-AFs und deutet auf eine ähnlich regelmäßige Struktur. %ia ist in ABA2AF-AFs mit $|A|$, $|R|$ und $|R|/|A|$ stark negativ korreliert. Planning2AF-AFs zeigen für %ia keine signifikanten Werte mit $|A|$, $|R|$ und $|R|/|A|$. Allerdings zeigt %pa* mit $|A|$, $|R|$, sowie $|R|/|A|$ eine schwache negative Korrelation. Mit steigender Größe der AFs sinkt eher der Anteil an pa* Argumenten, als dass er steigt. Zusammen mit der stark negativen Korrelation zwischen %ia und %pa erwarten wir, dass für %ia das Gegenteil, wenn auch abgeschwächt, gilt. Als letzten Punkt finden wir für Traffic-AFs eine moderat negative Korrelation zwischen %ia und $|R|/|A|$, aber keine signifikanten zwischen %ia und $|A|$ oder $|R|$ alleine.

Insgesamt sehen wir im pbbg_2040-Datensatz eine begrenzte und im iccma_450-Datensatz eine ermutigend große Bandbreite an unterschiedlichen Korrelationskoeffizienten für die Beziehungen von %ia zu den anderen Eigenschaften (abgesehen von %pa). Damit ist es unwahrscheinlich, dass wir zumindest für den iccma_450-Datensatz aus der Stärke einer bestimmten Korrelation. Falls dies nicht der Fall wäre, könnten wir versuchen mit der Kenntnis dieser Eigenschaft eines AFs wie $|A|$ die Eingangsmerkmale zu augmentieren und eine höhere Leistung zu erzielen oder einen naiven Argument-Klassifizierer wie [KWT22] statt nur mit dem durchschnittlichen Ein- und Ausgangsgrad von pa Argumenten auch zusätzlich diese Eigenschaft in die Entscheidung miteinfließen lassen. Wir finden solche starken bis sehr starken Korrelationen aber nur einzelne AF-Typen. Falls dementsprechend eines un-

[84] Vgl. allgemein Tabelle 6. Für diese und die folgenden Prozentangaben nutzen wir jedoch nicht die Daten aus der Tabelle, sondern unsere kaum gerundeten Originaldaten.

[85] Vgl. allgemein Tabelle 8, mit der selben Einschränkung wie in der vorherigen Fußnote.

serer Modelle sicher zwischen AF-Typen (und zwischen u. U. durch eine mögliche Eigenschafteninstabilität durch Skalierung induzierten Untertypen) unterscheiden könnte, könnten wir für viele dieser Typen spezifische nützliche Informationen über die Art und Stärke der Korrelationen von %ia/%pa zu einfach zu berechnenden AF-Eigenschaften wie $|A|$, $|R|$ und $|R|/|A|$ extrahieren (z. B. ist %ia zu $|R|/|A|$ in Barabási-Albert-Typen stark bis sehr stark negativ korreliert).[86] Wir begnügen uns hier erst einmal mit diesem Befund.

4.3 Repräsentative Abschlussdatensätze

Wir trainiern einem Teil eines Datensatzes ein Modell und überprüfen am Rest des Datensatzes die Qualität des Trainings. Je repräsentativer dieser Abschlusstestdatensatz für den gesamten Datensatz ist, desto aussagekräftiger sind unsere Ergebnisse. Bei einer rein zufälligen Auswahl ist es zum Beispiel nicht ausgeschlossen, dass ein bestimmter AF-Typ des Datensatzes weniger oder u. U. gar nicht nicht im Trainings- oder Abschlusstestdatensatz vertreten ist. Selbst wenn ein Typ in beiden Datensätzen vertreten ist, ist es dem Zufall überlassen, wie welche der AFs auf diese verteilt werden. So können sich die AFs in den Datensätzen, obwohl sie zu demselben Typ gehören, stark im Einfluss aufs Training (z. B. über eine unterschiedliche Einordnungschwierigkeit für KNNs) unterscheiden. So sind im Allgemeinen AFs eines Typs mit wenigen Argumenten einfacher lösbar als die mit sehr vielen Argumenten[87].

Die im Abschlusstest gemessene Leistung variiert dementsprechend, inwiefern zufällig mit den „passenden“ oder „falschen“ AFs trainiert und getestet wurde. Uns geht es nun nicht darum, eine möglichst hohe Leistung über Manipulationen der Datensätze zu erzielen, sondern einen möglichst stabilen und aussagekräftigen Abschlusstest durchzuführen. Hierfür möchten wir – erstens – garantiert alle AF-Typen eines Datensatzes in den Trainings- und Abschlusstestdatensätzen wiederfinden, – zweitens – das ungefähre Verhältnis der Anzahl der AFs pro AF-Typen erhalten und – drittens – AFs eines Typs so für den Abschlusstestdatensatz auswählen, dass die Auswahl eher die Gesamtheit der AFs eines Typs repräsentiert. Wir nutzen als Repräsentation eine Näherung bezüglich der drei Kriterien Gesamtzahl an Argumenten $|A|$, Durchmesser und dem Anteil an ia Argumenten %ia.[88]

[86] Wir vermuten, dass diese Klassifizierung der Typen mit anschließender Sonderbehandlung auch eine der Operationen ist, die unsere KNNs durchführen. Nur sind AF-weite Informationen wie $|A|$, $|R|$ und $|R|/|A|$ durch die lokalisierten und in Distanz begrenzten Agglomerationen unserer KNNs nicht für die einzelnen Argumente und ihre Zustandswerte direkt verfügbar, sondern müssten anders zur Verfügung gestellt werden.

[87] Vergleiche hierzu die Resultate der einzelnen AF-Typen mit kleineren AFs in $E(pbbg)$ mit den korrespondierenden größeren in $E(iccma)$ in den Abschnitten **??** und **??**. Selbst für die in ihren Eigenschaften bei Skalierung sehr stabilen Barabási-Albert-AFs sinkt die Leistung. Auch für AGNNs sehen wir eine schlechtere Leistung nach derselben Zahl an Nachrichtenübermittlungsschritten bei größeren AFs (siehe [CB20]: S. 1670).

[88] Wie die Korrelationsmatrizen im letzten Abschnitt zeigen, sind diese Kriterien nicht voneinander unabhängig, aber die Stärke der linearen Abhängigkeiten ist begrenzt. In unseren Vorabtests zeig-

Für die repräsentative Auswahl entnehmen wir zunächst einem Gesamtdatensatz M die Untermenge M_{typ}, die nur AFs eines Typs enthält. Dann berechnen wir die Quartile von M_{typ} bezüglich der drei Kriterien. Für jedes AF in M_{typ} notieren wir in einem geordneten 3-Tupel t_{AF}, in welche der durch die Quartile abgegrenzten Viertel das AF fällt. Über den Tupel weisen wir den AFs jeweils eines von vier Fächern (engl.: *buckets*) B_{0-3} zu, die lose[89] mit den Vierteln korrespondieren:

$$B(AF) = \begin{cases} B_x, & \text{falls } t_{AF} = (x,x,y) \bigvee t_{AF} = (y,x,x); \\ B_0, & \text{falls } t_{AF} = (0,1,3); \\ B_1, & \text{falls } t_{AF} = (0,1,2); \\ B_2, & \text{falls } t_{AF} = (1,2,3); \\ B_3, & \text{falls } t_{AF} = (0,2,3). \end{cases}$$

$$\text{Mit } t_{AF} = (q_0, q_1, q_2),\ q_0 \leq q_1 \leq q_2, \text{ und } q_0, q_1, q_2, x, y \in \{0,1,2,3\}.$$

B_0 und B_3 beinhalten eher AFs mit Außenseiter-Werten bezüglich der Mehrheit der Kriterien und B_1 und B_2 eher AFs mit Durchschnittswerten. Wir fügen jetzt AFs zu unseren Abschlusstestdatensatz hinzu, bis letzterer mindestens einen bestimmten Anteil p (meist ist $p = 0,1$) des Gesamtdatensatzes enthält. Wir entnehmen aus jedem Fach im Normalfall dabei zufällig `ROUNDUP` ($|M_{typ}| \times p/4$) AFs und fügen sie dem Abschlusstestdatensatz hinzu. Den Rest legen wir in den Trainingsdatensatz. Für den Sonderfall, dass die Anzahl an AFs für einen Typ besonders klein ist, wählen wir gezielter, ansonsten wäre die Anzahl der AFs dieses Typs im Trainingsdatensatz sehr gering. Falls $|M_{typ}| < 1/p$ wählen wir nur ein AF aus,[90] ist $1/p <= |M_{typ}| < 2/p$ wählen wir zwei aus.[91] Als erstes AF wählen wir eines, das Durchschnittlichkeit bezüglich der Kriterien repräsentieren soll, und entnehmen es dementsprechend aus dem größeren Fach von B_1 und B_2. Sind sie gleich groß, wählen wir zufällig. Falls beide leer sein sollten, wählen wir zufällig eines aus den nicht leeren Fächern.[92] Falls wir zwei AFs entnehmen sollen, wählen wir für das zweite AF eines mit stärkerem Ausnahme-Charakter und entnehmen es dem größeren der beiden Fächer B_0 und B_3. Sind beide gleich große, wählen wir zufällig. Sind beide leer, wählen wir zufällig aus den anderen Fächern.

---

ten die von uns ausgewählten drei Kriterien sowie $|A|$, $|R|/|A|$ und %ia eine ähnlich hohe Ergebnisstabilität.

[89] Hierbei geht selbstverständlich im Normalfall Information verloren. Schon das geordnete Tupel $t_{AF} = (0,1,3)$ steht für sechs verschiedene Fälle ein. Insgesamt bilden wir die 64 mögliche Fälle aus $\{0,1,2,3\}^3$ auf {0, 1, 2, 3} ab.

[90] Für $p = 0,1$ ist dies $|M_{typ}| < 10$. Dieser Fall tritt bei uns beim Typ AdmBuster im iccma_450-Datensatz ein.

[91] Für $p = 0,1$ ist dies $10 <= |M_{typ}| < 20$. Dieser Fall tritt bei uns bei den Grounded- und Sembuster-Typen im iccma_450-Datensatz ein.

[92] Dies ist in den Datensätzen pbbg_2040, iccma_450 und kwt_2000 für $p = 0,1$ oder $p = 0,2$ nicht der Fall.

Auswahl	repräsentativ	zufällig
Mediane der MCCs aus je 16 Läufen	0,8337, 0,8200, 0,8218, 0,8383, 0,8327, 0,8345, 0,8404, 0,8471, 0,837, 0,8388	0,8153, 0,8440, 0,8399, 0,8456, 0,8386, 0,8458, 0,8319, 0,8467, 0,8446, 0,8157
Arithm. Mittel	0,834	0,837
Median der 10 Mediane	0,836	0,842
Standardabweichung	**0,0082**	**0,0121**

Tabelle 9: Vergleichtest zwischen repräsentativer und zufälliger Wahl. Wir nutzen `GCN3LIN3` und den pbbg_2040-Datensatz. Wir listen die Mediane der MCCs von je zehn Testreihen auf, die jeweils aus 16 Läufen bestanden. Wie erhofft, zeigt sich ein großer Unterschied in der Standardabweichung von 0,0082 zu 0,0121.

Durch dieses Auswahlverfahren wird der Abschlusstestdatensatz im Vergleich zu einer komplett zufälligen Auswahl dem ursprünglichen Datensatz bezüglich der Kriterien mit größerer Wahrscheinlichkeit ähneln und damit repräsentieren. Wir erwarten, dass dies zu stabileren und aussagekräftigeren Ergebnissen führen. In einem Vergleichstest reduzierte diese Technik die Standardabweichung zwischen den MCC_{Median}-Werten um fast ein Drittel von 0,0121 auf 0,0082 (vgl. Tabelle 9). Wir führten dabei für die repräsentative und zufällige Auswahl je 10 Testreihen aus je 16 Testläufen durch und berechneten die Standardabweichung der 10 MCC_{Median}-Werte für jedes Auswahlverfahren.

5 Vortests, Experimente und Ergebnisse

Wir stellen in diesem Kapitel kurz die von uns benutzten Programme und Bibliotheken vor. Dann besprechen von uns durchgeführte Vortests und deren Auswirkung auf unsere Experimentalanordnung und anschließend die Experimentalanorndung selbst. Danach besprechen wir unsere Ergebnisse der drei Experimente $E(pbbg)$, $E(iccma)$ und $E(kwt)$.

5.1 Verwendete Programme

Zur Durchführung der Experimente nutzen wir eine Reihe von selbstgeschriebenen und externen Programmen. Für die Erstellung der AFs für den pbbg_2040-Datensatz nutzen wir zwei selbstgeschriebene Manager in Java für die Generatoren aus `probo` und `AFBenchgen`. Wir nutzen den Löser μ`-toksia` [NJ20], um unsere Grundwahrheiten zu erstellen. Ein dritter Manager in Java überwacht diesen Prozess und zeichnet die Lösungen auf. Wir erstellen Lösungen für alle[93] drei Daten-

[93] Für die beiden AFs belgium_2015.gml.20 und chicago_2016-01.gml.50 konnten wir wegen Zeitabläufen DS_{pr} für drei und für vier Argumente nicht berechnen.

sätze für SE_{id}, DC_{pr} und DS_{pr} und nutzen die Eigenschaft id$(AF) \subseteq$ pa$(AF) \subseteq$ pa*(AF) sowie die Bedingung der Konflikfreiheit zur Minimierung der nötigen Operationen. Wir berechnen zuerst id(AF) und wissen nun, welche Argumente ia und ina sind. Alle $a \in$ ia(AF) sind auch pa und pa*. Weiterhin wissen wir, dass jedes $b \in a^+ \cup a^-$ wegen der Bedingung der Konfliktfreiheit ina, pna und pna* ist. Jedes der restlichen Argumente kann entweder noch pa* und pa, pa* und npa, oder npa* und npa sein. Wir berechnen zunächst DC_{pr}, da es im Allgemeinen weniger rechenintensiv als DS_{pr} ist, und finden für die restlichen Argumente heraus, ob sie pa* oder pna* sind. Damit müssen wir nur noch für jedes $c \in$ pa*$(AF) \setminus$ id(AF) entscheiden, ob es pa oder npa ist. Hier können wir wieder Kandidaten wegen der Bedingung der Konfliktfreiheit aussschließen. Ein c ist sicher npa, falls $(c^+ \cup c^-) \cap$ pa*$(AF) \neq \emptyset$ ist. Für die c, für die $(c^+ \cup c^-) \cap$ pa*$(AF) = \emptyset$ ist, berechnen wir schließlich DS_{pr}.

Für die Verarbeitung der Daten im GNN nutzen wir ein an `TUDataset`[94] angelehntes Format aus Textdateien. Zur Zusammenführung der AFs und Grundwahrheiten sowie der Überführung in dieses Format nutzen wir einen vierten Manager in Java. In diesem Manager berechnet auch ein Unterprogramm für uns relevante Eigenschaften wie den Durchmesser über die Berechnung kürzester Wege. Der Manager speichert auch für die spätere Visualierung wichtige Informationen in csv-Dateien z. B. darüber, welche Ganzzahl-ID zu welchen AF-Typ gehört.

Die Programme für die Experimente und Datenanalyse sind in Python geschrieben und laufen in einer `Jupyter Notebook`-Umgebung. Wir nutzen die `PyTorch`-Bibliothek[95] und die `PyTorch Geometric`-Bibliothek[96] zur Implementation der GNNs und die `Pandas`-Bibliothek[97] zur Aufzeichnung und Analyse der Ergebnisse. Zur Erstellung der Abbildungen nutzen wir die Bibliotheken `Matplot`[98] und `seaborn`[99]. Für die Vortests und den kleineren Datensatz pbbg_2040 nutzen wir ein eigenes System (Intel Core i7-8086K, Nvidia RTX 2060 Super (8GB)) und für die größeren iccma_450 und kwt_2000 mieten wir uns Instanzen über `vast.ai` mit einer GPU mit mehr VRAM (AMD EPYC 7542, Nvidia RTX 3090 (24GB)). Zur Echtzeitüberwachung der Experimente nutzen wir neptune.ai.

5.2 Vortests

Wir führten eine Reihe von Vortests durch, um unsere Experimentalanordnung anzupassen. In den wichtigsten konnten wir im Vergleich zur bisherigen Forschung mit kleinen pbbg-Datensätzen feststellen, dass wir mit zwei Veränderungen – zum einen dem Weglassen der GCN-Normalisierung und zum anderen durch das Anschließen von linearen Schichten an die GCN-Schichten – die Erkennungsleistung von flachen und tiefen GCNs wesentlich erhöhen können. Diese Auswirkungen be-

[94] https://chrsmrrs.github.io/datasets/docs/format/
[95] https://pytorch.org
[96] https://pytorch-geometric.readthedocs.io
[97] https://pandas.pydata.org
[98] https://matplotlib.org
[99] https://seaborn.pydata.org

Design	Läufe	MCC: Median	MCC: CV	TPR	TNR	ACC
GNN8LIN5	5	0,885	**0,4**%	0,821	1,0	0,962
GNN8LIN5	16	0,891	1,7%	0,823	1,0	0,962
GNN8LIN5	48	0,895	1,4%	0,825	1,0	0,963
GNN8LIN5	80	0,891	1,0%	0,827	1,0	0,963
GNN8LIN5	128	0,886	**15,6**%	0,804	1,0	0,958
GNN8LIN5	160	0,889	5,0%	0,822	1,0	0,962

Tabelle 10: Vergleich der Kennzahlen nach einer verschiedenen Anzahl an Läufen von `GNN8LIN5` trainiert und getestet an pbbg_2040.

sprechen wir mit den Ergebnissen der Experimente. Zunächst beschreiben wir noch andere Tests, die unsere Experimentalanordnung beeinflussen.

5.2.1 Eingangsmerkmale

Wir testeten die Nutzung von verschiedenen Eingangsmerkmalen. Wir verglichen an einem kleinen pbbg-Datensatz und dem flachen `GCN2LIN0` die Leistung mit Eingangmerkmalen aus aus einem Zufallswert (wie in [KT19]),der Anzahl der ein-, sowie ausgehenden Angriffe (wie in [KT19]) und aus `DeepWalk`-Ergebnissen (wie in [MYNM20]). Zufallswertmerkmale zeigten die schlechtesten Ergebnisse, aber auch – für uns überraschend – die `DeepWalk`-Merkmale schnitten durchgehend schlechter als die Anzahl der ein- und ausgehenden Angriffe ab, weshalb wir uns für Nutzung letzterer entschieden haben.[100]

5.2.2 Variabilität und Kennzahlen

Bei vielen der anderen Vortests ist uns bei der Leistung von Modellen bestimmter Designs eine hohe Variabilität aufgefallen, die unabhängig von der Zahl der Trainingsepochen war. Wir entschieden uns dafür, diese Variabilität mitzuerfassen, statt sie durch frühes Stoppen o. ä. zu mitigieren. Pro Experiment und Design messen wir deshalb fünf Läufe mit fester Epochenzahl.[101] Zumindest an am Beispiel des lesitungsstarken Designs `GNN8LIN5` haben wir unsere Ergebnisse mit fünf Läufen mit denen mit 16, 48, 80, 128 und 160 Läufen vergleichen können und eine gute Übereinstimmung in den Kennzahlen gefunden. Wenngleich es nach 128 Läufen zu einem wesentlich mit höherem Median-CV-Wert kam (vgl. Tabelle 10).

Wir nutzen als Hauptkennzahl der Leistung eines Designs deshalb auch den Median der MCC-Werte der fünf Läufe und der Lauf des Medians steht für die Erken-

[100] Im Nachhinein bleibt für die Frage offen, inwiefern ein anderer Datensatz wie iccma_450 (der ein Teil der in [MYNM20] genutzten Datensätze B_L und B_T ist) andere Ergebnisse gezeigt hätte. Oder vielleicht hätten andere Designs andere Ergebnisse gezeigt.

[101] Wir nehmen nicht an, dass wir bei nur fünf Läufen die Stabilität oder Variabilität eines Designs in Gänze einfangen können. Eine wesentlich höhere Anzahl an Läufen war mit den uns zur Verfügung stehenden Rechenressourcen und -zeit nicht möglich gewesen.

nungsgenaugkeit des Designs ein. Eine neue Nebenkennzahl ist der Variationskoeffizient oder kurz CV (engl.: *coefficient of variation*) der MCCs der fünf Läufe. Der CV der MCCs fängt die relative Größe der Abweichungen ein, indem die Standardabweichung durch das arithmetische Mittel der MCC-Werte geteilt wird.

Anstatt des arithmetischen Mittels nutzen wir den kleinsten der beiden mittleren validen Werte, falls nur eine gerade Anzahl an Läufen einen validen MCC aufweisen, und unterschätzen damit im Zweifel die Erkennungsgenauigkeit. Das arithmetische Mittel zweier MCCs spiegelt nicht unbedingt den Leistungsmittelpunkt zwischen den Läufen wider. Eine andere Möglichkeit wäre es gewesen, die Konfusionsmatrizen der beiden Mittelwerte zu addieren (oder die Summe für alle Läufe zu bilden) und dann daraus den MCC zu berechnen. Aber damit würden wir MCC-Werte für virtuelle, nie durchgeführte Experimente berechnen, was wir hier vermeiden möchten. Falls keine MCCs für die Medianbildung nutzbar sind, und die ACC-Werte sich unterscheiden sollten, nutzen wir den Median der ACC-Werte und den dazugehörigen Lauf. Die weiteren Nebenkennzahlen wie TPR, TNR und ACC gehören zu dem Lauf, der den Median der MCCs stellt.

5.2.3 Gewichtung der Verlustfunktion

Die Datensätze pbbg_2040 und iccma_450 weisen mit einem Anteil an ia Argumenten von 22% und 9,9% eine unausbalanzierte Klassenverteilung auf. In solchen Fällen kann eine gewichtete Verlustfunktion die Erkennung der unterrepräsentierten Klasse verbessern, indem die Fehlzuordnung von Argumenten der unterrepräsentierten hier positiven Klasse stärker bestraft wird. Wir nutzten deshalb für einen Vortest `torch.nn.BCEWITHLOGITSLOSS` statt `torch.nn.BCELoss()`, den Datensatz pbbg_2040 und das Design `GNN4LIN3` (siehe Tabelle 11 auf Seite 57). Über die Gewichtung der Verlustfunktion werden die Kosten für die Fehleinordnung von ia Argumenten um den Faktor „Trainingsgewicht“ verändert. Für jedes Trainingsgewicht `pos_weight` führten wir 10 Läufe durch und notierten wie erläutert die Kennzahlen des Laufes des Medians der MCCs. Ein höheres Gewicht verbesserte stets die TPR, aber für die MCC-Werte waren die Ergebnisse nicht eindeutig. Bis `pos_weight` = 1,6 stiegen auch der Median der MCCs auf 0,846 und die ACC auf 0,949, bei höheren Gewichten nahmen beide aber wieder ab. Insgesamt führte das Training über die gewichtete Verlustfunktion selbst im besten Fall zu einer etwas schlechteren Erkennungsleistung als das Training über die ungewichtete. Mit der ungewichten Verlustfunktion erreichte `GNN4LIN3` einen Median der MCCs von 0,852 (+0,006) und eine ACC von 0,952 (+0,003). Wir haben uns deshalb gegen den Einsatz einer gewichteten Verlustfunktion entschieden. Für eine Optimierung bei besonders unausgewogenen Datensätzen ist sie aber wegen der Erhöhung der TPR interessant.

Design	Trainings-gewicht	MCC: Median	MCC: VC	TPR	TNR	ACC
GNN4LIN3$_{1,0}$	1,0	0,837	14,3%	0,756	0,998	0,947
GNN4LIN3$_{1,25}$	1,25	0,840	3,1%	0,758	**0,999**	0,947
GNN4LIN3$_{1,6}$	1,6	**0,846**	19,5%	0,780	0,995	**0,949**
GNN4LIN3$_{2,0}$	2,0	0,836	6,7%	0,812	0,983	0,946
GNN4LIN3$_{2,5}$	2,5	0,823	16,2%	**0,818**	0,976	0,942
GNN4LIN3	1,0	0,852	3,8%	0,805	0,992	0,952

Tabelle 11: Vergleich der Ergebnisse von `GNN4LIN3` trainiert und getestet am pbbg_2040-Datensatz mit einer gewichteten Verlustfunktion und verschiedenen Gewichten und der ungewichteten Verlustfunktion.

5.2.4 Übertragbarkeit

Um festzustellen, ob und, wenn ja, inwiefern wir unsere Ergebnisse wegen der unterschiedlichen Problemstellung mit den bisherigen Forschungsergebnissen vergleichen können, verglichen wir in einem Vergleichstest verglichen wir die Leistung von sechs Designs bei SE_{id}, DS_{pr} und DC_{pr}. Im Abschnitt zur Analyse der Datensätze haben wir schon gezeigt, dass für die von uns genutzten Datensätzen nur 0,1% bis 0,3% der Argumente der Datensätze pa und ina sind; pa* und ina sind hingegen 10,6% bis 29,2% der Argumente. Wir erwarteten deshalb eine hohe Übertragbarkeit von SE_{id} zu DS_{pr} und nur eine begrenzte von SE_{id} zu DC_{pr}.

Wir zeigen nun die Übertragbarkeit für die acht Designs `GCN2LIN0`, `GCN2LIN3`, `GCN3LIN5`, `GCN5LIN0`, `GNN2LIN0`, `GNN4LIN1`, `GNN5LIN0` und `GNN8LIN3` (siehe Tabelle **??**.[102] Wir sehen für SE_{id} und DS_{pr} sehr ähnliche Ergebnisse, die sich nur in einem Design stärker unterscheiden: `GNN4LIN1`, das sich – wie wir später noch zeigen werden – durch einen besonders niedrigen Median der MCCs auszeichnet. Für DS_{pr} ist aber auch der Wert hier auch im Vergleich zu den anderen niedrig. Wir sehen bei den Werten der Designs mit KIPF-WELLING-normierten Schichten geringere Abweichungen als bei denen mit nicht normierten Schichten. Aber die Leistung der Designs in SE_{id} und DS_{pr} ähneln sich sehr. Für SE_{id} und DC_{pr} sehen wir größere Differenzen. So zeigt `GNN5LIN0` für DC_{pr} einen hohen Wert von 0,757 im Vergleich zu den 0,556 für SE_{id} und der Leistungsunterschied zwischen `GCN2LIN3` und `GCN3LIN5` sinkt von 0,101 für SE_{id} auf 0,029 für DC_{pr}. Damit ist der direkte Vergleich zwischen Ergebnissen für SE_{id} und DS_{pr} eingeschränkt möglich, der Vergleich mit DC_{pr}-Ergebnissen aber nur allgemein möglich. Für den pbbg_2040-Datensatz bleibt auch festzuhalten, dass in unserer Stichprobe – für uns überraschend – DS_{pr} die höchsten und DC_{pr} die niedrigsten Mediane der MCCs zeigt (vom Ausreißer `GNN4LIN1` abgesehen).

[102] Diese Designs waren nicht die, an denen wir den Vortest durchgeführt haben. Wir nutzen nun diese acht, da sie die Übertragbarkeit (und ihre Grenzen) zeigen, aber auch relevant für $E(pbbg)$ sind.

	SE_{id}	DS_{pr}		DC_{pr}	
Design	MCC: Median	MCC: Median	Diff.	MCC: Median	Diff.
GCN2LIN0	0,408	**0,423**	+0,015	0,398	-0,010
GCN2LIN3	0,719	**0,733**	+0,014	0,731	+0,012
GCN3LIN5	**0,820**	0,816	-0,004	0,760	-0,060
GCN5LIN0	**0,594**	0,591	-0,003	0,514	-0,080
GNN2LIN0	0,473	**0,512**	+0,039	0,487	+0,014
GNN4LIN1	0,197	**0,328**	+0,131	0,277	+0,080
GNN5LIN0	0,556	0,554	-0,002	**0,757**	**+0,201**
GNN8LIN3	0,878	**0,885**	+0,007	0,775	-0,103

Tabelle 12: Vergleich der Leistung von acht GNN-Designs für SE_{id}, DS_{pr} und DC_{pr} trainiert und getestet an pbbg_2040. Neben den Medianen der MCCs geben wir für die beiden letzten Probleme auch die Differenz zu SE_{id} an.

5.3 Experimentalanordnung

Für jedes der Experimente vergleichen wir die Erkennungsleistung von 108 Designs über je fünf Läufe. Die GNNs haben entweder KIPF-WELLING-GCN-Schichten oder nicht normierte Schichten, besitzen zwischen zwei und zehn GCN-Schichten gefolgt von keiner bis fünf linearen Schichten. Wir nutzen eine Lernrate von 0,001, unsere versteckten Schichten haben 64 Dimensionen und wir nutzen die Anzahlen der eingehenden und ausgehenden Angriffe als Eingangsignale für jedes Argument. Wir trainierten mit dem Adam-Optimierer und `Binary Cross-Entropy` als Verlustfunktion. Wir brechen den Lernprozess nach 200 Epochen ab und nutzen die dann aktuellen Parameter als Modell. Wir nutzen das *Dropout*-Verfahren mit einer Wahrscheinlichkeit von 0,2 und inkludieren Argumente mit einer Trainingswahrscheinlichkeit von 0,8 pro Epoche ins Training. Wir entnehmen aus jedem der Datensätze pbbg_2040, iccma_450 und kwt_2000 repräsentativ bezüglich der Kriterien $|A|$, Durchmesser und %ia je einen Abschlusstestdatensatz und benutzen den Rest als Trainingsdatensatz. Damit ist der Rahmen unserer Experimente $E(pbbg)$, $E(iccma)$ und $E(kwt)$ gegeben.

Bei der Besprechung der Ergebnisse jedes Experiments nutzen wir zuerst eine nach der Höhe des Medians der MCCs sortierte Liste der Designs, in der auch alle Nebenkennzahlen aufgelistet sind. Jede dieser Listen ist zweieinhalb Seiten lang und deshalb im Anhang verortet. Wir teilen diese Liste in grobe Bereiche, in denen sich die Leistung und/oder die Designs geändert haben. Dann besprechen wir das Abschneiden des Designs `GCN2LIN0`, das `FM2` sehr ähnelt, und die der res-GCNs etwas ähnlichen `GCN4LIN0`, `GCN5LIN0` und `GCN6LIN0`. Wenn wir dabei die Ergebnisse der anderen Experimente mit unseren vergleichen, versuchen wir uns entweder allgemein zu halten oder nur Ähnliches zu vergleichen. So nutzen wir dieselben oder ähnliche Datensätze wie KUHLMANN, WUJEK und THIMM in [KWT22] und die Ergebnisse des von ihnen genutzten Problems DS_{pr} sind nach unserer Stichprobenanalyse mit unseren für SE_{id} vergleichbar (mit einem etwas schelchteren Ergeb-

nissen für SE_{id}). Nach den Vergleichen mit Designs und Ergebnissen der bisherigen Forschung analysieren wir den Einfluss von Normierung, GCN-Schichtenzahl und linearer Schichtenzahl auf die MCC-Ergebnisse noch einmal genauer über zwei Tabellen mit neun Zeilen und sechs Spalten: eine Tabelle mit den MCC-Median-Ergebnissen der Designs mit normierten und eine Tabelle mit den nicht normierten Designs. Diese Tabellen sind nach Anzahl der GCN-Schichten und nach Anzahl der linearen Schichten geordnet. Wir suchen in diesen Tabellen nach Mustern, die in der Listenform untergehen. Für Testdatensätze aus pbbg_2040 und iccma_450 analysieren wir weiterhin die Ergebnisse bezüglich der AF-Typen in weiteren Tabellen.

5.4 Experiment $E(pbbg)$

In diesem Experiment trainieren und testen wir die Designs am pbbg_2040-Datensatz. Er ist in Bezug auf $|A|$ der kleinste unserer Datensätze, weshalb wir 20% statt 10% der Graphen repräsentativ für den Abschlusstestdatentest reservieren, um aussagekräftigere Ergebnisse zu erhalten. Dies sind 68 pro AF-Typ.

5.4.1 Analyse der Liste

Als erste Beobachtung folgen MCC[103] und ACC derselben Reihenfolge bis auf den untersten Bereich (vgl. Tabelle 14 im Anhang auf Seite ii). Die Designs mit der höchsten Leistung sind allesamt nicht normiert und haben eine hohe kombinierte Schichtenanzahl. Sie bilden für uns den oberen Bereich von Stelle 1 bis 31. Den Höchstwert mit einem MCC von 0,891, einer TPR von 0,831, einer TNR von 1,0 und einer ACC = 0,964 erreicht `GNN8LIN3`. Die ersten zwanzig Designs liegen nahe beieinander mit MCC-Werten von 0,872 oder höher. Das erste KIPF-WELLING-normierte Design `GCN3LIN5` steht an Platz 32 von 108 mit einem MCC von 0,82. Hier lassen wir den mittleren Werte-Bereich beginnen. Er besteht aus fast nur normierten Designs und endet mit einschließlich `GCN7LIN5` an Stelle 73 mit einem MCC-Wert von 0,77. Alle diese Designs schlagen den von `res-GCN4` mit dem balanzierten Datensatz $B_T(bal)$ erreichten bisherigen MCC-Höchstwert eines GCNs von 0,636. Im letzten Bereich (Stelle 74 bis 108) fällt die Erkennungsleistung stärker ab. Wir teilen ihn in zwei Unterbereiche. Der erste Unterbereich bis Stelle 88 besteht vor allem aus Designs mit nur zwei GNN-Schichten mit MCC-Werten zwischen 0,733 und 0,559. Im zweiten Unterbereich ab der 89. Stelle finden wir Werte von 0,557 bis – (nein) und nur Designs mit keiner oder – seltener – mit einer linearen Schicht. Viele der nicht normierten Designs zeigen hier eine hohe Variabilität mit CV-Werten bis zu 59,3%. Andere Modelle scheinen mit CV-Werten von 0,0% in denselben suboptimalen lokalen Maxima gefangen. Die Designs mit der geringsten Leistung sind fünf normierte Nein-Entscheidern und erreichen damit ein MCC-Äquivalent von

[103] Wir werden den Begriff des Medians der MCC-Werte der fünf Läufe eines Designs in den folgenden Abschnitten der detaillierten Besprechung der Experimente sehr oft nutzen, so dass wir ihn aus Gründen der Lesbarkeit im Fließtext zu MCC abkürzen und stets anmerken, falls dies nicht der Fall ist.

0,0. Keines hat eine lineare Schicht und alle haben mehr als fünf GCN-Schichten. Insgesamt überrascht das gute Abschneiden der besonders tiefen GNNs mit generalisierten Schichten. Der Einfluss der linearen Schichten ist auch unter den zehn besten Designs deutlich: Unter ihnen befinden sich fünf Designs mit fünf und zwei Designs mit vier linearen Schichten, aber kein Design ohne und nur eines mit einer linearen Schicht.

Das mit `FM2`-Pendant `GCN2LIN0` kommt an 101. Stelle und erreicht einen MCC von 0,408 bei geringer Variabilität, was geringer als der Wert 0,558 im korrespondierenden Experiment (mit etwas kleineren AFs) $D_1(D_{L1}, D_{T1})$ ist. Wegen dieses Unterschieds gehen wir davon aus, dass die Argumente der AFs unseres pbbg-Datensatzes schwieriger einzuordnen sind als die in Fall D genutzten. Schauen wir uns weiter dazu die Nebenkennzahlen an, finden wir in unseren Tests eine im Vergleich zu seinen Stellennachbarn hohe TPR von 0,612 und eine niedrige TNR von 0,828. Die TPR und TNR ähneln denen in $D_1(D_{L1}, D_{T1})$. Dort wurde beim Training und Testen mit pbbg-AFs eine etwas höhere TPR von 0,67 und etwas höhere TNR von 0,882 erreicht. Die ACC ist mit 0,782 niedrig und liegt unter der eines Nein-Entscheiders mit ACC = 0,786. Die Designs, die eine Zwischenschritt zwischen res-GCNs und `FM2` bilden, zeigen unterschiedliche Ergebnisse. `GCN5LIN0` und `GCN4LIN0` erreichen höhere Stellen als das `FM2`-Pendant und folgen an Stelle 84 und 85 aufeinander mit im Vergleich zu den GCN-Ergebnissen der bisherigen Forschung recht hohen MCC-Werten von 0,594 und 0,588 und unauffälligen Nebenkennzahlen. `GCN6LIN0` hingegen teilt sich den letzten Platz und alle seine Modelle sind Nein-Entscheider. Von einem direkten Vergleich mit den Ergebnissen der res-GCNs der Experimente B_1 oder B_2 sehen wir hier ab, da sich die genutzten Datensätze zu stark voneinander unterscheiden.

Mit unserem MCC-Höchstwert von 0,891 liegen wir deutlich über dem Wert von 0,558, den `FM2` in $D_1(D_{L1}, D_{T1})$ erreicht, aber auch deutlich unter den 0,962 die ein AGGN in derselben Experimentalanordnung erreicht. Wir vermuten, dass ein Teil dieses Unterschieds in den Datensätzen begründet ist, da auch unser `FM2`-Pendant mit 0,408 einen wesentlich schlechteren Wert zeigt. So sind unsere AFs nicht nur größer, sondern haben auch mit 22,0% statt 28,3% einen geringeren Anteil an Argumenten in der positiven Klasse, was im Allgemeinen das Lernen und Einordnen erschwert (siehe die zugehörigen Einträge in den Tabellen 4 und 6).

5.4.2 Basisanalyse nach Normierung und Schichtenzahlen

Ordnen wir die MCC-Werte nach Normierung, Anzahl der GCN-Schichten und Anzahl der linearen Schichten erhalten wir zwei Tabellen mit je neun Zeilen und sechs Spalten (vgl. Abbildung 6 auf Seite 61). Beginnen wir mit der Tabelle der normierten Designs. Hier zeigen Designs mit keiner linearen Schicht wesentlich schlechtere Ergebnisse, selbst wenn wir die Nein-Entscheider ignorieren. Im Mittel liegt ihr MCC um 0,245 unter dem der Designs mit einer linearen Schicht. Auch Designs mit nur zwei GCN-Schichten schneiden schlechter ab und liegen im Mittel um 0,096 unter

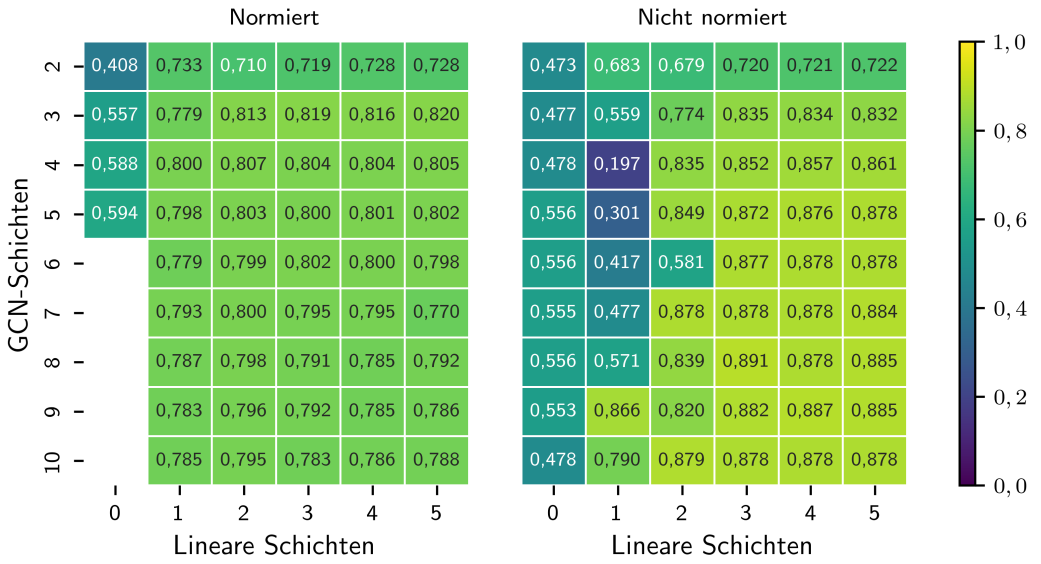

Abbildung 6: Die Mediane der MCC-Werte der verschiedenen Designs für das gesamte Experiment $E(pbbg)$ sortiert nach Normierung, Anzahl der GCN-Schichten und Anzahl der linearen Schichten.

dem Wert von Designs mit drei GCN-Schichten. Ansonsten zeigen die Designs mit drei und mehr GCN-Schichten und einer oder mehr linearen Schichten unter sich kaum Unterschiede mit Werten um 0,8 bei einer leichten Verschlechterung mit zunehmenden GCN-Schichten. Die höchsten Werte haben die Designs mit drei GCN-Schichten und zwei und mehr linearen Schichten mit Werten von 0,813 und darüber mit dem Maximum bei `GCN3LIN5` mit 0,82.

In der Tabelle der nicht normierten Designs zeigen auch die Designs ohne lineare Schicht und die Designs mit nur zwei GCN-Schichten schlechtere Ergebnisse als die anderen Designs. Diesmal treten keine Nein-Entscheider auf. Allerdings zeigt sich ein interessantes Leistungsloch bei einigen Designs mit genau einer linearen Schicht. `GNN3LIN1` zeigt mit einem MCC von 0,197 eine wesentlich niedrigere Erkennungsleistung als jedes andere normierte Design inklusive des Minimums bei `GNN2LIN0` mit 0,473. Auch `GNN2LIN1`, `GNN4LIN1`, `GNN5LIN1` und `GNN6LIN1`, sowie `GNN6LIN2` zeigen auffällig niedrige Werte im Vergleich zu ihren Nachbarn, so dass wir den Leistungsabfall nicht als zufälliges Phänomen betrachten. Diese Gruppe and Designs bildet in unserer Tabelle zusammen ein keilförmiges Muster. Diese Designs zeichnen sich auch durch eine hohe Variabilität aus, was zusammen auf ein unbekanntes Trainingsproblem dieser spezifischen Designs zumindest beim pbbg_2040-Datensatz hindeutet. Ansonsten sind die meisten MCC-Werte im Vergleich zu den normierten Architekturen mit über 0,83 gegenüber über 0,79 relativ hoch. Sie stabilisieren sich aber erst bei höherer Schichtenzahl auf hohem Niveau. Eine „Insel der Stabilität“ finden wir hier nicht schon bei drei GCN- und einer linea-

ren Schicht wie bei den normierten Designs, sondern erst bei mehr als vier GNN-Schichten und mehr als zwei linearen Schichten. Hier sind alle MCC-Werte höher als 0,87 mit dem Maximum des Experiments bei `GNN8LIN3` mit 0,891.

5.4.3 Analyse nach Normierung und Schichtenzahlen pro AF-Typ

Filtern wir das Ergebnis nach AF-Typen, so bleiben die Beobachtungen im Großen erhalten (vgl. Abbildungen 7 (Seite 63) und 8 (Seite 64)). Der Bereich normierter Designs mit drei und mehr GCN-Schichten und mindestens einer linearen Schicht zeigt gleichbleibend hohe MCC-Werte. Bei nicht normierten Designs existiert zusätzlich noch ein keilförmiger Bereich niedriger Werte und der Bereich gleichförmig hoher Werte beginnt erst bei tieferen GNNs, erreicht aber höhere Maxima als bei den normierten.

Wir analysieren zuerst die drei AF-Typen mit den höchsten MCC-Werten und versuchen die Ergebnisse auf Eigenschaften der Typen zurückzuführen. Die Argumente der Barabási-Albert-AFs lassen sich sehr gut von unseren Designs korrekt einordnen. Die große Mehrheit der normierten Design zeigt sehr hohe MCC-Werte von über 0,92, und 23 davon zeigen Werte von 0,95 und höher. Bei den nicht normierten Designs zeigt nur etwas die Hälfte der Designs hohe Werte, aber fast alle von denen sind 0,95 und höher mit 0,97 als Höchstwert. 15 der Designs zeigen sogar einen perfekten MCC. Ähnliche Schwellenwerte finden wir auch für die anderen AF-Typen, wobei wir bei den anderen AF-Typen niedrigere Maximalwerte finden. Die zweitbesten MCC-Werte finden wir in Grounded-AFs. Hier zeigt die Mehrheit der normierten Designs aber nur MCC-Werte von knapp unter 0,8 und darüber und die Spitzenwerte von über 0,84 stellen vier Designs mit drei GCN-Schichten und zwei und mehr linearen Schichten mit einem Höchstwert von 0,859. Bei den nicht normierten Designs zeigt die Hälfte MCC-Werte von über 0,88 und 19 davon mit Werten von 0,95 und darüber mit dem Höchstwert von 0,977. Die nächstbesten Werte finden wir Für Stable-AFs. Eine Mehrheit der normierten Designs hat einen MCC von über 0,74, und die höchsten Werte mit 0,787 und darüber haben erneut die vier Designs mit drei GCN-Schichten und zwei und mehr linearen Schichten mit einem Höchstwert von diesmal 0,790. Bei den nicht normierten Designs hat die Hälfte einen MCC von 0,815 und höher mit fünf davon mit Werten von 0,87 und darüber bei einem Höchstwert von 0,878.

Wenn wir die Datenanalyse der Eigenschaften der AF-Typen rekapitulieren, sind in Barabási-Albert-AFs 41,5% der Argumente ia, in Grounded-AFs sind es 31,3% und in Stable-AFs sind es 17,8%. Aufgrund unserer Vortests erwarten wir, dass eine höhere Ausgeglichenheit eines Datensatzes die Einordnung leichter macht. Dies ist hier der Fall. Allerdings hätten wir wegen der Größe des Sprungs von Grounded zu Stable von 31,3% auf 17,8% ein heftigeres Abfallen des MCCs erwartet. Ein anderer Einfluss auf die Lösungsschwierigkeit ist die Anzahl der Angriffe pro AF. Wir gehen auch davon aus, dass weniger Angriffe im Allgemeinen eine Lösung erleichtern. Für unsere AF-Tyoen finden wir im Median niedrige 54 Angriffe pro AF bei Barabási-

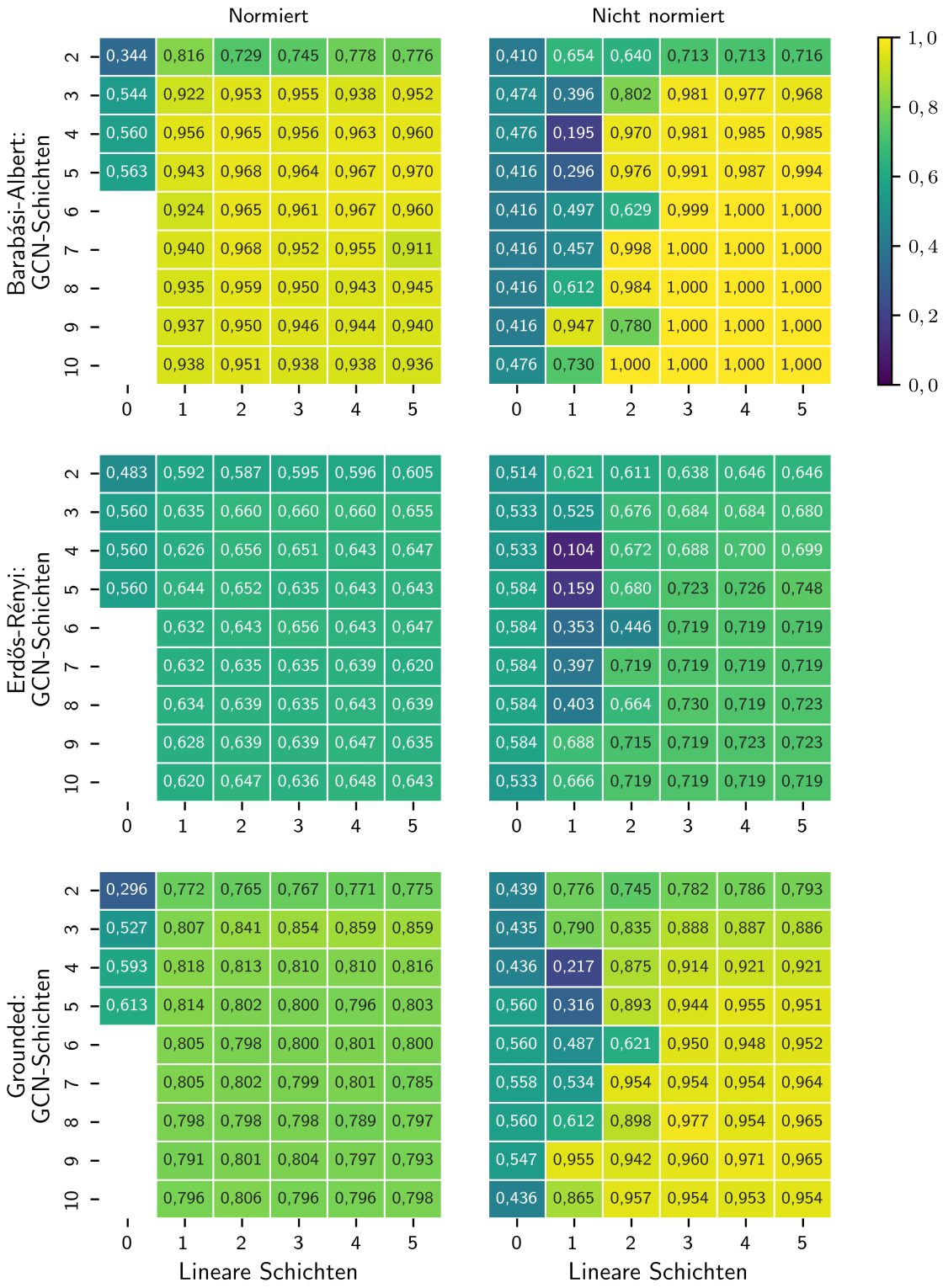

Abbildung 7: Die Mediane der MCC-Werte der AF-Typen Barabási-Albert, Erdős-Rényi und Grounded des Experiments $E(pbbg)$ sortiert nach Normierung, Anzahl der GCN-Schichten und Anzahl der linearen Schichten.

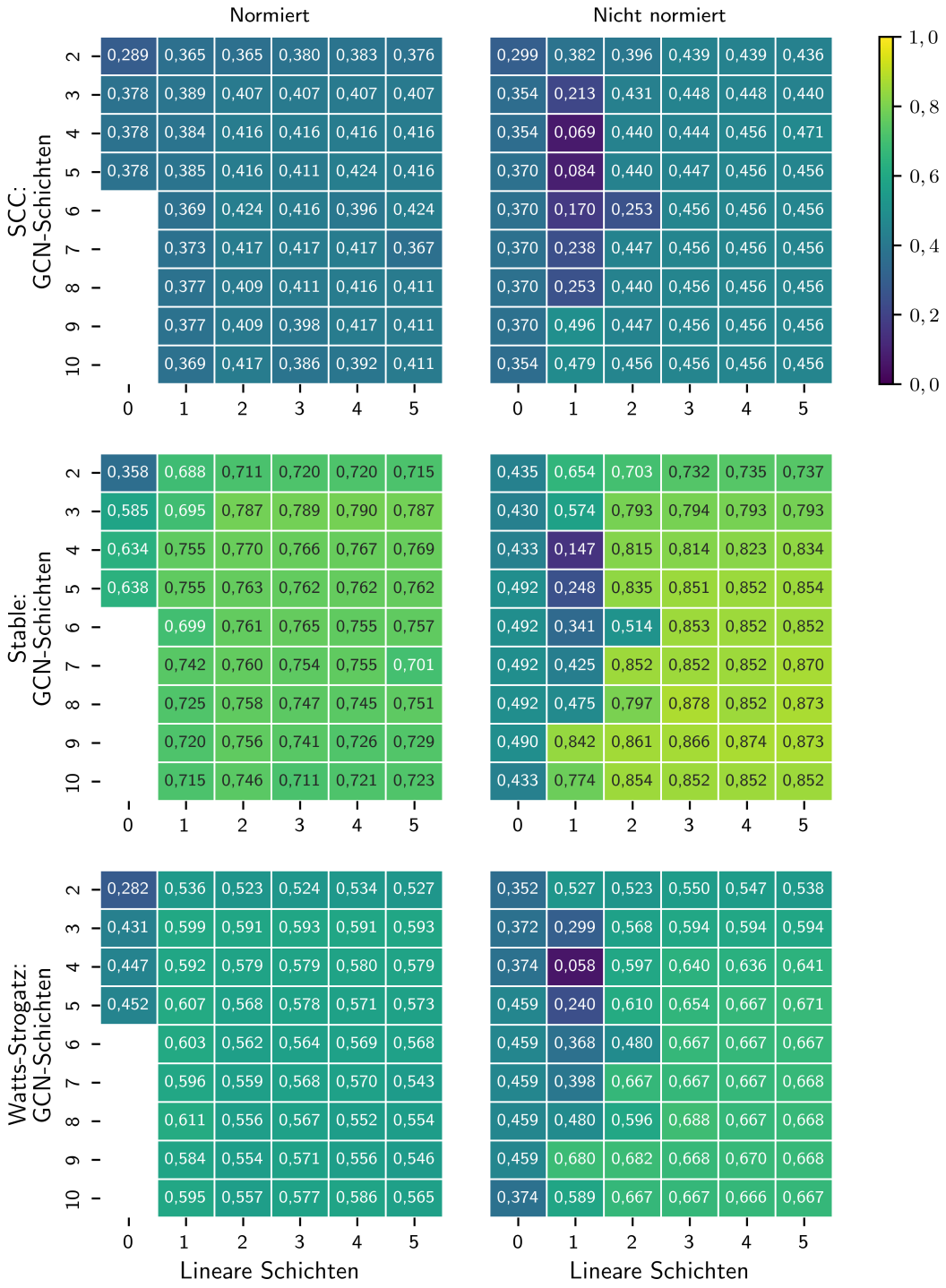

Abbildung 8: Die Mediane der MCC-Werte der AF-Typen SCC, Stable und Watts-Strogatz von $E(pbbg)$.

Albert-AFs, aber relativ 136 bei Grounded- und moderate 83 bei Stable-AFs. Der letzte Unterschied kann eine Erklärung für die Begrenzung des Abfallens sein.

Betrachten wir nun die letzten drei Typen verkürzt. Bei Erdős-Rényi-AFs zeigen die meisten normierten Designs MCCs von über 0,63 mit einem Höchstwert von 0,66, viele der nicht normierten Designs zeigen Werte über 0,71 mit einem Höchstwert von 0,748. Bei Watts-Strogatz-AFs zeigen die meisten normierten Designs einen MCC von über 0,55 und einen Höchstwert von 0,611. Hier sind als Ausnahme die Designs mit genau einer linearen Schicht leistungsstark. Bei den nicht normierten Designs haben viele einen MCC von über 0,65 bei einem Höchstwert von 0,688. Die niedrigsten MCC-Werte finden wir bei SCC-AFs. Die meisten normierten Design haben hier einen MCC von über 0,4 bei einem Höchswert von 0,424 und die meisten nicht normierten Designs haben einen Wert von 0,44 und darüber und einen Höchstwert von 0,496 (überraschenderweise bei einem Design mit nur einer linearen Schicht direkt am Rande des keilförmigen Bereichs niedriger Leistung). Hier passt die Reihenfolge der Leistung nicht mit der Reihenfolge von %ia zusammen. Nur 7,8% der Argumente in den Erdős-Rényi-AFs sind ia, gegenüber 17,5% in Watts-Strogatz- und 6,0% in SCC-AFs. Auch haben Erdős-Rényi-AFs mit einem Median von 311 Angriffen pro AF ähnlich viele wie SCC-AFs mit 292 und wesentlich mehr als Watts-Strogatz-AFs mit 100. Wir können uns diese Reihenfolge mit keiner der üblichen Eigenschaften erklären, weshalb sie dem Typ inhärent sein muss. Damit sind zumidnest kleine Erdős-Rényi-AFs im Verhältnis zu ihren Eigenschaften einfacher zumindest für unsere Designs zu lösen als die ihnen in den Eigenschaften ähnlichen SCC-AFs.

Der Unterschied zwischen dem höchsten MCC eines normierten und dem höchsten MCC eines nicht normierten Designs ist bei Barabási-Albert-AFs mit 0,03 am geringsten (`GCN5LIN5` und `GNN6LIN5`) und bei Grounded-AFs mit 0,118 (`GCN3LIN4` und `GNN8LIN3`) am höchsten. Für Grounded-AFs scheint die Gerichtetheit der Angriffsrelation eine wichtige Information zu sein. Der Unterschied liegt sonst bei den anderen Typen zwischen 0,066 und 0,088. Damit gibt es keinen AF-Typen im pbbg_2040-Datensatz, den normierte Designs besser lösen.

5.5 Experiment $E(iccma)$

Im Experiment $E(iccma)$ trainieren und testen wir Designs am iccma_450-Datensatz. Wie wir im Abschnitt **??** gezeigt haben, ist dieser Datensatz sehr inhomogen. So besteht z. B. die Untergruppe der AdmBuster-AFs nur aus fünf AFs, stellt damit nur 1,1% der AFs des Datensatzes aber 7,9% der Argumente und sogar 39,6% der ia Argumente. Für dieses Experiment haben wir mit 61 Graphen etwas mehr als 13% der AFs repräsentativ in den Abschlusstestdatensatz gelegt.

5.5.1 Analyse der Liste

Die MCC-Reihenfolge stimmt dieses Mal nur im obersten Bereich mit der ACC-Reihenfolge überein (vgl. Tabelle 15 auf Seite iv). Wiederum sind die Designs mit

der höchsten Leistung nicht normiert. Dieses Mal folgt aber schon an 14. Stelle ein normiertes Design. Wir lassen an dieser Schwelle den oberen Bereich enden. Diese ersten dreizehn Designs haben MCC-Werte von 0,903 bis 0,791 und haben erneut eine hohe summierte Schichtentiefe von 9 bis 15, sowie in der Mehrheit vier oder fünf linearen Schichten. Zunächst überraschend schlägt der höchste MCC-Wert von 0,903 dabei knapp den des letzten Experiments von 0,891, dafür sinken die MCC-Werte mit zunehmendem Rang schneller. Vergleichen wir die zehn besten Architekturen mit denen von $E(pbbg)$, dann fällt auf, dass vor allem die TPR rapide abnimmt. Hier fällt die TPR von Stelle 1 zu 10 von 0,848 auf 0,755, während sie in $E(pbbg)$ nur von 0,831 auf 0,811 abnimmt. Der Unterschied in der Varianz über den CV-Wert fällt mit im Mittel 9,8% im Vergleich zu 0,57% in $E(pbbg)$ deutlich aus. Beide Befunde zusammen deuten auf einen weniger stabilen Lernprozess für den iccma_450-Datensatz, selbst wenn wir die jeweils leistungsstärkstens zehn Designs vergleichen.

Im folgenden Bereich von Stelle 14 bis Stelle 60 folgen normierte und nicht normierte Designs und wir finden keine weitere sinnvolle strukturelle oder Kennzahlenbezogene Unterteilung. Die besten normierten Designs in diesem Bereich sind tief und haben alle zehn oder mehr summierte Schichten, mit dem besonders tiefen `GNN10LIN4` als leistungsstärkstem normierten Design mit einem MCC von 0,776. Wir lassen den unteren Bereich ab Stelle 61 beginnen. Er beginnt mit einigen nicht normierten Designs mit keiner oder einer GCN-Schicht (61.-72.) mit im Mittel sehr geringer Varianz, gefolgt von gemischten Designs mit hoher Varianz (73.-98., CV bis zu 71,7%), und endet abschließend mit neun Nein-Entscheidern (99.-108.), wovon drei nicht normierte Designs sind. Den schlechtesten Wert mit einem negativen MCC von -0,002 zeigt `GNN7LIN1`. Die Nein-Entscheider haben wegen der Unausgeglichenheit der Klassen des Datensatzes eine hohe ACC von 0,906.

Unser `FM2`-Pendant `GCN2LIN0` zeigt die schlechteste ACC aller Designs mit 0,576. Sie hat auch einen niedrigen MCC von 0,247, dafür eine recht hohe TPR von 0,879 und eine sehr niedrige TNR von 0,544. Dieses Experiment stellte also `GCN2LIN0` vor große Schwierigkeiten. Der MCC von 0,247 ist unerwartet niedrig. Er ist nur ein wenig höher als der Wert von 0,211, der im Experiment $D_1(D_{L1}, D_{T3})$ beim Trainieren mit einem pbbg_2040-ähnlichem Datensatz und einem Testen am ganzen iccma_450-Datensatz durch `FM2` erzielt wurde, also in einem Test, bei dem viele der AF-Typen (und Untertypen) dem Modell unbekannt waren.

Die res-GCN-ähnlichen Designs `GCN5LIN0` und `GCN6LIN0` sind Nein-Entscheider, nur `GCN4LIN0` hat mit 0,216 einen validen, wenn auch noch niedrigen MCC-Wert als unser `FM2`-Pendant. Der Wert erscheint dennoch hoch, wenn wir die Nebenkennzahlen betrachten. Die TPR liegt mit 0,051 nahe null und die TNR bei 1,0, was `GCN4LIN0` fast zu einem Nein-Entscheider macht. Die ACC ist wegen der Unausgeglichenheit des Datensatzes mit 0,911 recht hoch. Da wir mit einem Teil des ICCMA-2017-Datensatzes trainieren und testen, können wir nun unsere Ergebnisse mit denen aus B_1 und B_2 vergleichen. Das Problem von B_1 ist DC_{pr} und der schlechteste MCC-Wert von 0,4 von `res-GCN6` liegt weit über unserem einzigen validen Wert eines verwandten Designs von 0,216. In B_2 ist das Problem DS_{pr} und hier übertreffen

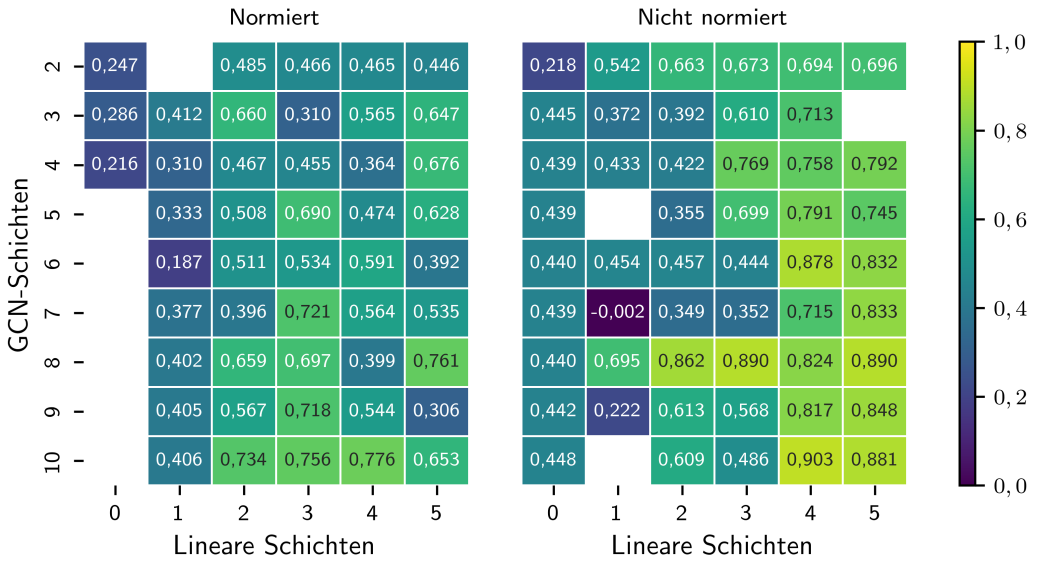

Abbildung 9: Die Mediane der MCC-Werte der verschiedenen Designs für das gesamte Experiment $E(iccma)$ sortiert nach Normierung, Anzahl der GCN-Schichten und Anzahl der linearen Schichten.

wir den höchsten MCC-Wert von `res-GCN4` von 0,1326. Aber unsere TPR von 0,051 ist so niedrig im Vergleich zu der von 0,24 von `res-GCN4`, dass wir unser Design für das leistungsschwächere halten. Alle normierten GCNs ohne lineare Schichten haben mit dem Datensatz große Probleme, die mit der Zahl der GCN-Schichten zunehmen (mit `GCN3LIN0` als Ausreißer). Erst andere Designs mit linearen Schichten und/oder ohne Normalisierung übertreffen die Ergebnisse in B_1 und B_2 deutlich.

5.5.2 Basisanalyse nach Normierung und Schichtenzahlen:

Betrachten wir nun die MCC-Werte für normierte und nicht normierte Designs bezüglicher ihrer Schichten genauer (siehe Abb. 9 auf Seite 67). Wir beginnen mit den normierten Designs. Auf den ersten Blick sind die Werte niedriger und weniger klar geordnet im Vergleich zum letzten Experiment $E(pbbg)$. Die Ergebnisse von Designs mit zwei GCN-Schichten sind etwas niedriger als die von denen mit drei GCN-Schichten. Dieser Unterschied ist u. a. wegen des Leistungsabfalls bei `GNN3LIN3` aber weniger stark als im vorherigen Experiment. Designs ohne lineare Schichten haben nun nur drei statt vier valide MCC-Werte und im Mittel einen wesentlich niedrigeren Wert von 0,25 statt 0,537 in $E(pbbg)$. Die Ergebnisse der Designs mit einer linearen Schicht fallen mit im Mittel 0,354 gegenüber 0,782 in $E(pbbg)$ ähnlich niedrig aus. Erst Designs mit drei und mehr linearen Schichten haben in der Mehrheit MCC-Werte von über 0,5 mit einigen Ausreißern nach oben und wenigen nach unten. Einem stabile, mehrere Spalten und Zeilen umfassende Region mit

grundsätzlich hohen Werten ist nicht zu finden. Nur Designs mit zehn GCN- und zwei oder mehr linearen Schichten, bzw. sieben oder mehr GCN-Schichten bei genau drei linearen Schichten erreichen hohe und stabile MCC-Werte über 0,65. Genau hier liegt auch bei `GCN10LIN4` das Maximum der normierten Werte mit 0,776. Wegen der hohen Varianz ist es ohne weitere Tests schwierig, abschließend zu urteilen, inwiefern diese Reihe und Spalte höherer Leistung Produkte des Zufalls sind oder stabile Muster.

Betrachten wir nun die Ergebnisse der nicht normierten Designs. Wir finden keinen deutlichen regionalen Leistungsabfall um `GNN3LIN1` herum, allerdings ragt die keilförmige Region niedriger Leistung tiefer in die Tabelle hinein und beinhaltet auch Designs mit drei linearen Schichten. Außerdem finden wir drei Nein-Entscheider in `GNN5LIN1`, `GNN10LIN1` und `GNN3LIN5`. Die MCC-Werte von Designs ohne lineare Schichten ist mit im Schnitt 0.417 wesentlich höher als bei normierten Designs, auch wenn die Leistung von `GNN2LIN0` im Vergleich zu den anderen stark abfällt. Die Designs mit genau einer linearen Schicht zeigen sehr unterschiedliche MCC-Werte von sehr niedrigen -0,002 bis hohen 0,695. Erneut zeigen nicht normierte Designs mit nur einer linearen Schicht keine deutliche Leistungstendenz. Generell hoch sind die Ergebnisse der Design mit vier und fünf linearen Schichten nur mit einem nicht validen Wert Ausreißer. Dies ist anders als bei den normierten Designs. Haben die normierten Designs auch noch acht und mehr GCN-Schichten haben, zeigen sie sehr hohe MCC-Werte von 0,817 bis 0,903. Auch hier finden wir eine Reihe mit hohen MCC-Werten mit den Designs mit genau 8 GNN-Schichten.

5.5.3 Analyse nach Normierung und Schichtenzahlen pro AF-Typ:

Filtern wir das Ergebnis nach Typen, finden wir überraschende Ergebnisse, die wir im Gegensatz zu $E(pbbg)$ detaillierter betrachten möchten. Wir beginnen mit den Ergebnissen für Erdős-Rényi-, Sembuster- und Watts-Strogatz-AFs, für die wir keine Tabelle benötigen. Beim Erdős-Rényi-Typ ist in unserem Abschlusstestdatensatz das einzige AF des Typs, das ia Argumente besitzt (ER_300_30_9.tgf mit $|ia| = 18$). Alle unsere Designs werden hier zu Nein-Entscheidern und ordnen jedes seiner ia Argumente durchgehend falsch ein.[104] Wie im Abschnitt **??** beschrieben haben diese AFs in ihren Eigenschaften wenig mit denen vom selben Typ aus dem pbbg_2040-Datensatz gemein.

Bei AFs vom Sembuster-Typ ist kein Argument ia (oder pa), damit sind keine validen MCC-Werte möglich. Bis auf das `GNN2LIN0`, das einige der ina Argumente falsch einordnet (TNR beträgt nur 0,999 statt 1,0), werden hier alle anderen Designs zu Nein-Entscheidern und sagen die Lösung korrekt voraus.

[104] Ein möglicher Grund dafür ist, dass diese vier Argumente äußerst schwierig einzuordnen sind. Unser Lösungsprogramm μ`-toksia` [NJ20] brauchte für die Lösung dieses AFs mit 789,4s länger als für jedes andere Erdős-Rényi-AF. Einige andere Erdős-Rényi-AFs zeigten auch lange Lösungsdauern, so belief sich der Durchschnitt der Lösungszeiten auf 137,3s und der Median nur auf 54,7s. Wir halten die Lösung dieses AFs dementsprechend für anspruchsvoll aber nicht außergewöhnlich schwierig.

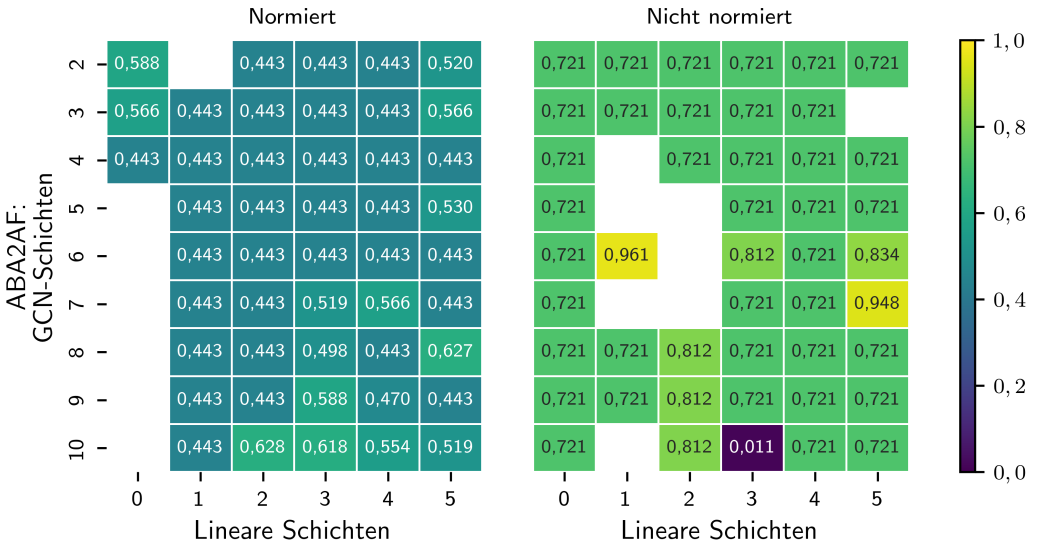

Abbildung 10: Die Mediane der MCC-Werte für den AF-Typ ABA2AF des Experiments $E(iccma)$ sortiert nach Normierung, Anzahl der GCN-Schichten und Anzahl der linearen Schichten.

Kein AF vom Typ Watts-Strogatz, das ia Argumente enthält, ist Teil unseres Abschlusstestdatensatzes[105]. Damit existieren hier ebenfalls keine validen MCC-Werte. Alle unsere Designs werden hier zu Nein-Entscheidern und lösen die Einordnung korrekt.

ABA2AF: Für ABA2AF-AFs (siehe Abb. 10) zeigen die meisten normierten Designs genau einen MCC-Wert: 0,443, der wesentlich unter dem unseres `FM2`-Pendants `GCN2LIN0` mit 0,588 liegt. Einige andere Designs liegen im Zwischenbereich. Nur drei besonders tiefe GCNs bilden mit höheren MMCs die Ausnahme: `GCN8LIN5` mit 0,627, `GCN10LIN2` mit 0,628 und `GCN10LIN3` mit 0,618.

Die nicht normierten Designs zeigen in der Mehrheit ebenfalls genau einen MCC-Wert: 0,721. Dieser liegt um 0,093 höher als der des besten normierten Designs. Acht Designs sind Nein-Entscheider, sieben davon liegen in der Keilstruktur. Die zwei Designs `GNN6LIN1` und `GNN7LIN5` zeigen mit 0,948 und 0,961 äußerst gute Werte, stechen aber isoliert aus ihrer Nachbarschaft hervor.

AdmBuster: Bei den normierten Designs finden wir für die AFs vom AdmBuster-Typ entweder nicht valide MCC-Werte und Werte nahe 0,0 oder sehr gute werte über 0,987 (siehe Abb. 11). Kein Design ohne lineare Schichten hat einen validen MCC-Wert. Die meisten davon sind wie sonst üblich Nein-Entscheider. Aber `GCN2LIN0` und `GCN3LIN0` sind Ja-Entscheider. Beide Entscheider-Typen sind für

[105]Hier könnte eine Verbesserung unseres Auswahlalgorithmus ansetzen: Falls für einen Typ im Abschlusstestdatensatz zu wenig AFs mit ia Argumenten vorhanden sind, sollte ein Ausgleichsmechanismus greifen, der die Chance auf valide MCC-Werte erhöht.

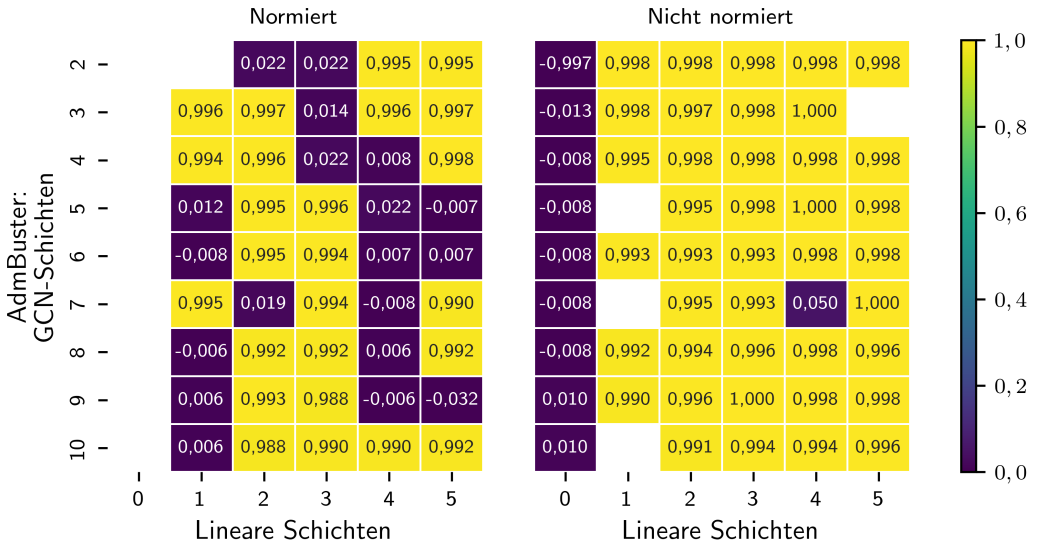

Abbildung 11: Die Mediane der MCC-Werte für den AF-Typ AdmBuster des Experiments $E(iccma)$ sortiert nach Normierung, Anzahl der GCN-Schichten und Anzahl der linearen Schichten.

diesen AF-Typ mit seiner balanzierten Klassenverteilung von ia zu ina gleichwertig. `GCN2LIN1` ist auch ein Nein-Entscheider. 19 der anderen Architekturen zeigen Werte um 0,0 herum[106]. Die anderen Designs zeigen hervorragende MCC-Werte von 0,988 bis zu 0,998. Wegen der vielen Ausreißer nehmen wir davon Abstand, andere Muster aus der Tabelle herauszulesen.

Bei den nicht normierten Designs fällt zuerst der hohe negative Wert -0,997[107] von `GNN2LIN0` auf. Fast jedes Argument wird inkorrekt eingeordnet, was bedeutet, dass das Modell zwar die Argumente korrekt in zwei Klassen teilen kann, aber dann die Klassen inkorrekt zuordnet. Auch die anderen nicht normierten acht Designs ohne lineare Schichten zeigen auffällige Werte. Hier liegen sie aber nahe 0,0 und die Designs verhalten sich alle ähnlich Nein-Entscheidern[108]. Es treten auch vier nicht valide MCC-Werte auf (drei davon bei Designs mit genau einer linearen Schicht), die zu Nein-Entscheider gehören. Die anderen Werte sind bis auf einen Ausreißer bei `GNN7LIN4` mit 0,99 bis 1,0 sehr hoch bis zu perfekt und schon flache Designs mit einer linearen Schicht wie `GNN2LIN1` zeigen sehr gute Werte mit 0,998.

[106] Hiervon verhalten sich sieben wie Nein-Entscheider mit äußerst niedriger TPR und zwölf wie Ja-Entscheider mit äußerst niedriger TNR. Zusammen mit den tatsächlichen Ja- und Nein-Entscheidern ergibt sich ein ausgeglichenes Verhältnis und weder die Ja- noch die Nein-Tendenz wird bevorzugt. Während bei den meisten anderen AF-Typen durch die Unausgeglichenheit Ja-Entscheider im Lernprozess stark benachteiligt werden, da sie wesentlich höhere Verluste haben, zeigen sie sich hier unbetroffen.

[107] `GNN2LIN0` zeigt hier auch eine hohe Konstanz mit einem CV von 0,0% über die fünf Läufe.

[108] Dies deutet darauf hin, dass die nicht normierten Designs anders als die normierten lernen.

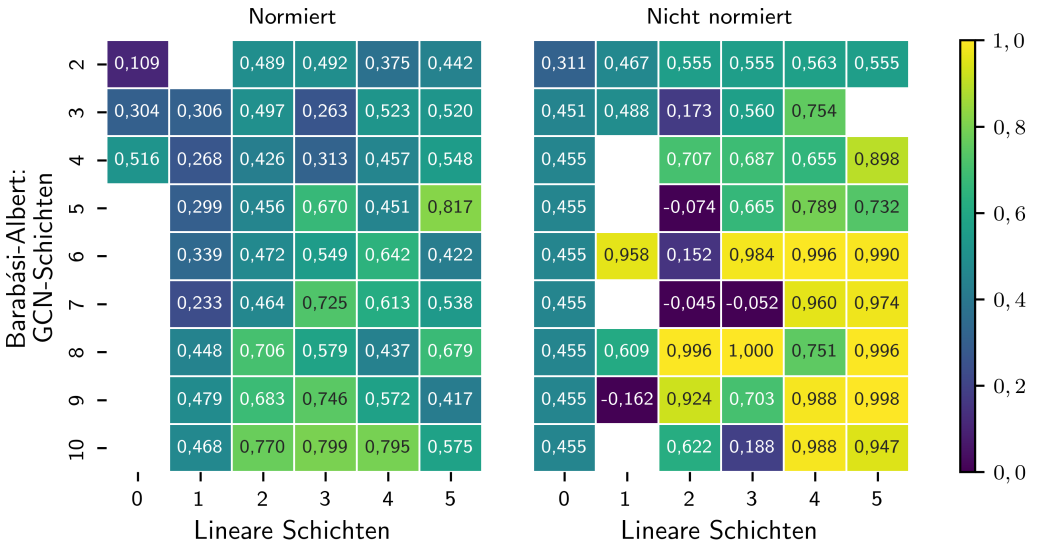

Abbildung 12: Die Mediane der MCC-Werte für den AF-Typ Barabási-Albert des Experiments $E(iccma)$ sortiert nach Normierung, Anzahl der GCN-Schichten und Anzahl der linearen Schichten.

Damit können wir Modelle für AdmBuster-AFs nach ihrem Einordnungsverhalten in distinkte Gruppen einteilen, zwischen denen keine graduellen Zwischenformen existieren, in denen die TPR und TNR Zwischenwerte annimmt. Entweder das Modell ‚versteht' die AdmBuster-Struktur (im Fall von `GNN2LIN0` versteht es falsch), oder nicht. Dies steht im Gegensatz zum MCC-Wert von -0,454 den wir in $D_2(kwt, AdmBuster)$ sehen.

Barabási-Albert: Bei den normierten Designs (siehe Abb. 12) sind erneut alle ohne lineare Schichten und mit mehr als 4 GCN-Schichten Nein-Entscheider. Das `FM2`-Pendant `GCN2LIN0` hat mit 0,109 einen sehr geringen MCC-Wert und verhält wegen einer TPR von 1,0 und TNR von 0,028 erneut ähnlich wie ein Ja-Entscheider. Die Barabási-Albert-AFs in unserem Abschlusstestdatensatz zeigen auch eine eher ausgeglichene Klassenverteilung mit 41% ia Argumenten. Das einzige valide Ergebnis eines mit res-GCNs verwandten Design ist wieder `GCN4LIN0` mit wesentlich besseren 0,516 als `GCN2LIN0`. Vor allem die Designs mit nur einer linearen Schicht sowie – schwächer ausgeprägt – die mit wenigen GCN-Schichten haben niedrige MCC-Werte. Hohe Werte finden wir nur in direkter Nachbarschaft um `GCN10LIN3` (alle direkten Nachbarn zeigen Werte über 0,74) sowie als höchsten Wert isoliert 0,817 bei `GCN5LIN5`. Im Vergleich mit den Werten für Barabási-Albert-AFs im Experiment $E(pbbg)$ fällt das Fehlen der gleichförmigen Verteilung sehr hoher Werte auf.

Bei den nicht normierten Designs zeigen die Designs ohne lineare Schichten nur moderate Werte. Viele der Designs mit einer linearen Schicht sind Nein-Entscheider, eines zeigt schwach negative Werte und die anderen moderate Werte. Nur `GNN6-`

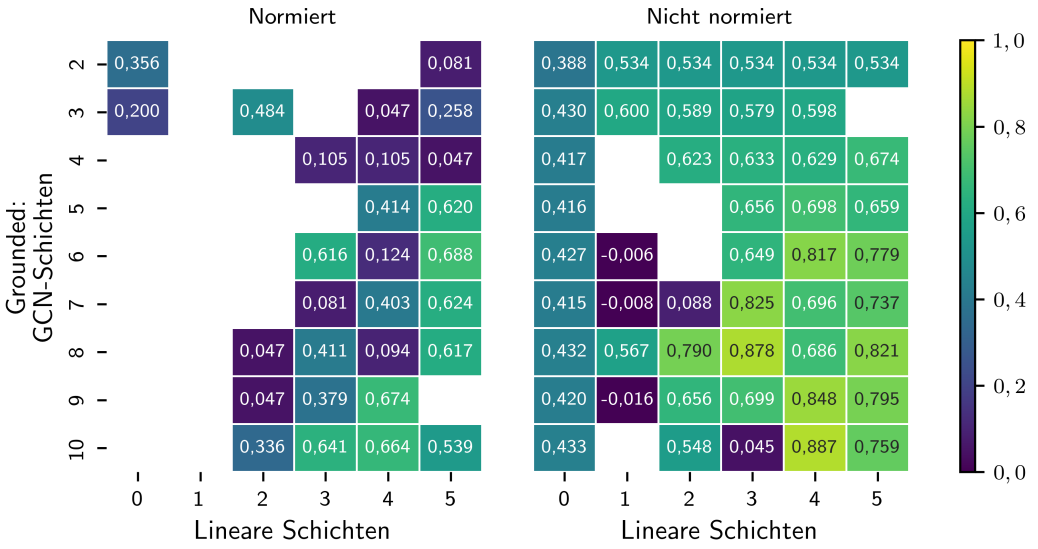

Abbildung 13: Die Mediane der MCC-Werte für den AF-Typ Grounded des Experiments $E(iccma)$ sortiert nach Normierung, Anzahl der GCN-Schichten und Anzahl der linearen Schichten.

`LIN1` sticht mit 0,958 (wie bei den ABA2AF-AFs) positiv heraus. Vor allem im Bereich der Keilform finden wir weitere Werte nahe 0,0. In dem Bereich mit Designs mit sechs und mehr GCN-Schichten sowie mit vier oder mehr linearen Schichten finden wir nur sehr hohe Werte über 0,94 von einem Ausreißer mit 0,751 abgesehen. Das dem Bereich benachbarte Design `GNN8LIN3` hat sogar einen perfekten Wert von 1,0.

Im Vergleich mit den Werten für Barabási-Albert-AFs im Experiment $E(pbbg)$ fällt auch bei nicht normierten Designs die fehlende Regelmäßigkeit auf. Eventuell ist der von uns identifizierte Bereich hoher Werte der Beginn eines größeren Bereichs, der ähnlich gleichförmig wie der in $E(pbbg)$ ist, nur eben erst bei höheren Schichtzahlen ansetzt.

In $E(pbbg)$ war der Unterschied zwischen den Höchstwerten der normierten und nicht normierten Design mit +0,003 sehr gering. Hier ist er mit 0,183 hoch.

Grounded: Fast alle normierten Designs mit geringer Schichtensumme werden bei Grounded-AFs zu Nein-Entscheidern (siehe Abb. 13). Nur `GCN2LIN0` und `GCN3LIN0` sowie `GCN3LIN2` haben in dem breiten Streifen, der sich in einem Bogen von `GCN10LIN1` bis `GCN2LIN4` zieht, niedrige, aber valide MCC-Werte. Die MCCs vieler Designs, die direkt an der Grenze des Bereichs liegen, sind nahe 0,0 und die Designs verhalten sich fast wie Nein-Entscheider. Erst mit zunehmender GCN-Tiefe und vor allem linearer Tiefe stabilisieren sich die Werte von einigen Ausreißern abgesehen auf über 0,6. Die Grounded-Ergebnisse von $E(pbbg)$ sind mit unseren Ergebnissen hier nicht zu vergleichen.

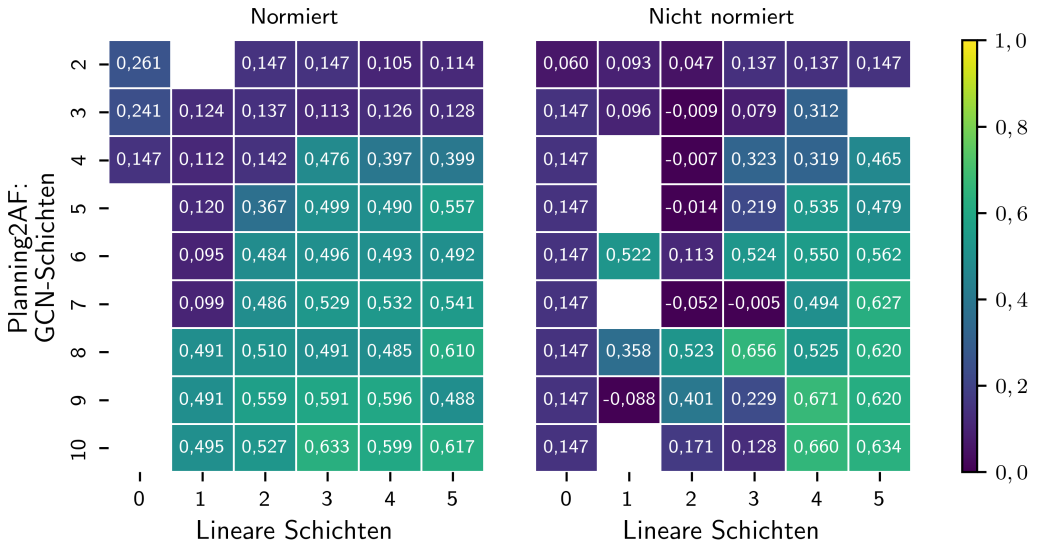

Abbildung 14: Die Mediane der MCC-Werte für den AF-Typ Planning2AF des Experiments $E(iccma)$ sortiert nach Normierung, Anzahl der GCN-Schichten und Anzahl der linearen Schichten.

Bei den nicht normierten Architekturen finden wir wesentlich mehr valide Werte. Die nicht validen oder schwach validen Werte finden wir in Architekturen mit nur einer linearen Schicht, die zum Großteil entweder direkte Nein-Entscheider sind oder sehr niedrige MCC-Werte haben und sich Nein-Entscheider-ähnlich verhalten, und in den drei Designs mit zwei linearen und drei, vier sowie fünf GCN-Schichten. In demselben Bereich, in dem Barabási-Albert-AFs stabil hohe Werte zeigen, finden wir hier auch hohe Werte über 0,736. Dieses Mal kommt zu `GNN8LIN4` mit nur 0,686 aber noch der zweite Ausreißer `GNN7LIN4` mit 0,696 hinzu. Der direkte Nachbar `GNN8LIN3` zeigt erneut einen hohen Wert mit 0,878. Damit zeigen die nicht normierten Designs hier sehr ähnliche Strukturen in Bezug zu denen der Barabási-Albert-AFs, wohingegen die normierten etwas komplett Neues zeigen. Im Vergleich zu $E(pbbg)$ erreichen wir nicht dieselbe Leistung und die Region hoher Leistung ist wesentlich kleiner und weniger gleichförmig.

Planning2AF: Bei den normierten Designs sehen wir zum ersten Mal bei AFs vom Typ Planning2AF einen deutlichen Vorteil von tiefen Designs ((siehe Abb. 14). `GCN2LIN0` und `GCN3LIN0` zeigen mit 0,261 und 0,241 niedrige Werte, die aber über den sehr niedrigen MCC-Werten der anderen Designs mit zwei oder drei GCN-Schichten liegen, die stets unter 0,15 bleiben. `GCN4LIN0` reiht sich unter die niedrigen Werte mit 0,147 ein. Die anderen Designs ohne lineare Schichten sind Nein-Entscheider. Designs mit einer linearen Schicht und weniger als acht GCN-Schichten haben niedrige MCC-Werte von unter 0,15. Dasselbe gilt für `GCN4LIN2`. Wir sehen wie bei Grounded-AFs wieder einen Bogen-förmige Bereich geringer Leistung. Die

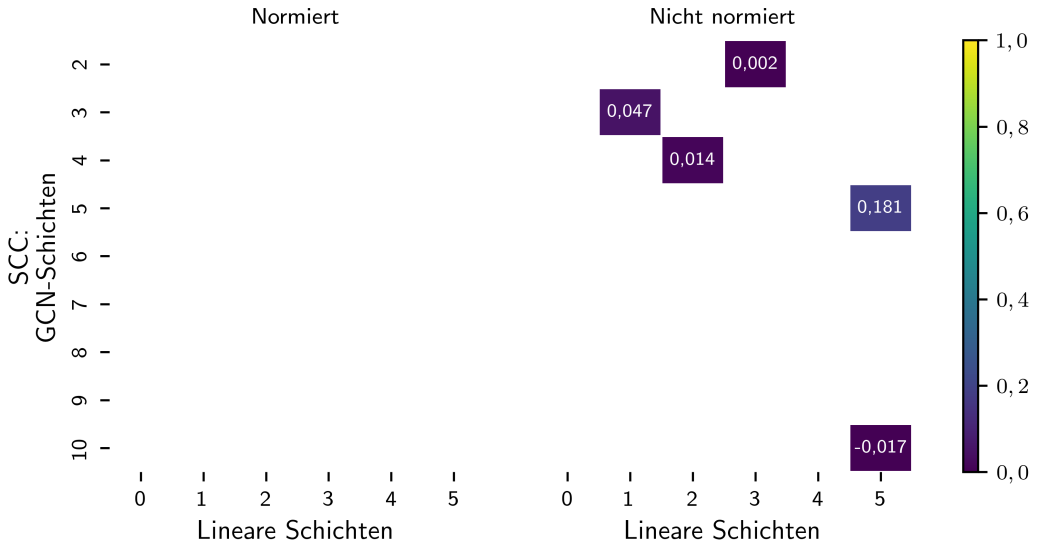

Abbildung 15: Die Mediane der MCC-Werte für den AF-Typ SCC des Experiments $E(iccma)$ sortiert nach Normierung, Anzahl der GCN-Schichten und Anzahl der linearen Schichten.

direkten Nachbarn dieser Architekturen haben höhere MCC-Werte von 0,367 bis 0,495. Die 23 anderen Designs, die alle eher tief sind, zeigen noch höhere Werte von 0,485 bis 0,633. Die Hälfte der tiefsten dieser Designs mit dreizehn oder mehr summierten Schichten erreicht dabei moderat hohe MCC-Werte von 0,61 bis 0,633.

Bei nicht normierten Designs stabilisieren sich die Werte diesmal erst bei hoher Schichtenzahl und auch hier ist ein bogenförmiger Bereich niedriger Leistung erkennbar. Die Designs mit keiner linearen Schicht zeigen sehr niedrige Wert von unter 0,15, diejenigen mit einer linearen Schicht sind zum großen Teil Nein-Entscheider (erneut mit `GNN6LIN1` als positiven Ausreißer) und diejenigen mit zwei linearen Schichten zeigen in der Mehrheit niedrige bis sehr niedrige MCC-Werte. Die Ergebnisse der Designs mit drei linearen Schichten sind gemischt. Erst ab vier linearen Schichten und fünf oder mehr GCN-Schichten sehen wir eine Stabilisierung. Die sechs besonders tiefen Designs `GNN9LIN4`, `GNN10LIN4`, `GNN7LIN5`, `GNN7LIN5`, `GNN8LIN5`, `GNN9LIN5` und `GNN10LIN5` haben alle moderat hohe MCC-Werte von 0,62 bis 0,671. Letzterer Höchstwert ist zwar etwas höher als der beste Wert der normierten Architekturen, allerdings fällt hier der Unterschied mit 0,038 im Vergleich zu den anderen AF-Typen im iccma_450-Datensatz gering aus.

SCC: Alle normierten Architekturen sind hier Nein-Entscheider (siehe Abb. 15). In unserem Abschlusstestdatensatz befindet sich ein einziges SCC-AF, das ia Argumente enthält, und zwar genau vier Argumente. Eine Fehleinschätzung dieser vier Argumente bei gleichzeitiger richtiger Einschätzung aller anderen hat auf den summierten Verlust für SCC-AFs eine so geringe Auswirkung, dass wir die positive

Design	MCC: Median	TPR	TNR	ACC
GNN2LIN3	0,002	1,0	0,003	0,004
GNN3LIN1	0,047	1,0	0,709	0,710
GNN4LIN2	0,014	0,5	0,710	0,710
GNN5LIN5	0,181	0,5	0,994	0,993
GNN0LIN5	-0,017	0,0	0,759	0,758

Tabelle 13: Vergleich der Kennzahlen, die zu den Läufen der Mediane der MCC-Werte gehören, die für SCC-AFs in $E(iccma, icmma)$ valide Werte aufweisen.

Klasse kaum für unser Training nutzen können. Um so überraschender, dass einige der nicht normierten von 0,0 abweichende MCC-Werte aufweisen, auch wenn sie auf den ersten Blick sehr gering sind. Da die eine Klasse aber nur vier Elemente beinhaltet und die andere mit 4430 um drei Größenordnungen mehr, untersuchen wir die Nebenkennzahlen der fünf Designs mit validen MCC-Werten genauer (siehe Tabelle 13 auf Seite 75).

Wie wir in unseren Anmerkungen zum MCC ausführten (siehe Abschnitt 2.4), führt eine solche extreme Unausgewogenheit zu MCC-Werten, die unseren Erwartungen an einen Kennwert, der die Unausgewogenheit von z. B. ACC ausgleicht nicht erfüllt. Wir finden, dass sich die MCC-Werte außer von `GNN2LIN3` im Betrag zu niedrig „anfühlen“, wenn wir die anderen Kennzahlen miteinbeziehen. Vor allem der Wert 0,181 von `GNN5LIN5` bei einer TPR von 0,5, einer TNR von 0,994 und einer ACC von 0,993 ist sehr niedrig. Bei einer ausgeglichenen Klassenverteilung von 4430:4430 und derselben TPR und TNR läge der MCC bei 0,568 und die ACC läge bei niedrigeren 0,747. Alle Designs mit validen MCC-Werten haben übrigens eine schlechtere ACC als die Nein-Entscheider. Aus Perspektive der Verlustfunktion ist es besser, keines der vier ia Argumente und dafür alle 4430 ina Argumente korrekt einzuordnen, als wie `GNN5LIN5` zwei der vier ia Argumente korrekt einzuordnen und nur 4403 der ina Argumente. Da ein Nein-Entscheider hier bevorzugt wird, ist es möglich, dass Trainingseinflüsse von AF anderer Typen zu diesen Ausreißern geführt haben. Ein Vergleich mit den SCC-AFs aus $E(pbbg)$ zeigt ein ganz anderes Verhalten. Aber die SCC-AFs aus iccma_450 zeigen auch insgesamt andere Eigenschaften als die pbbg_2040-SCC-AFs, wie wie in unserer Datensatzanalyse gezeigt haben.

Stable: Bei AFs diesen Typs sind die normierten Designs mit keiner oder einer linearen Schicht bis auf `GCN2LIN0` und `GCN3LIN0` Nein-Entscheider (siehe Abb. 16). Die restlichen Designs zeigen gemischte Ergebnisse. Jene mit höherer Tiefe weisen höhere MCC-Werte auf, aber oft unterbrochen durch Ausreißer nach unten. Designs mit zehn GCN-Schichten und drei oder mehr linearen Schichten haben durchgehend moderat hohe MCC-Werte von 0,501 bis 0,54 und stellen auch das Leistungsmaximum.

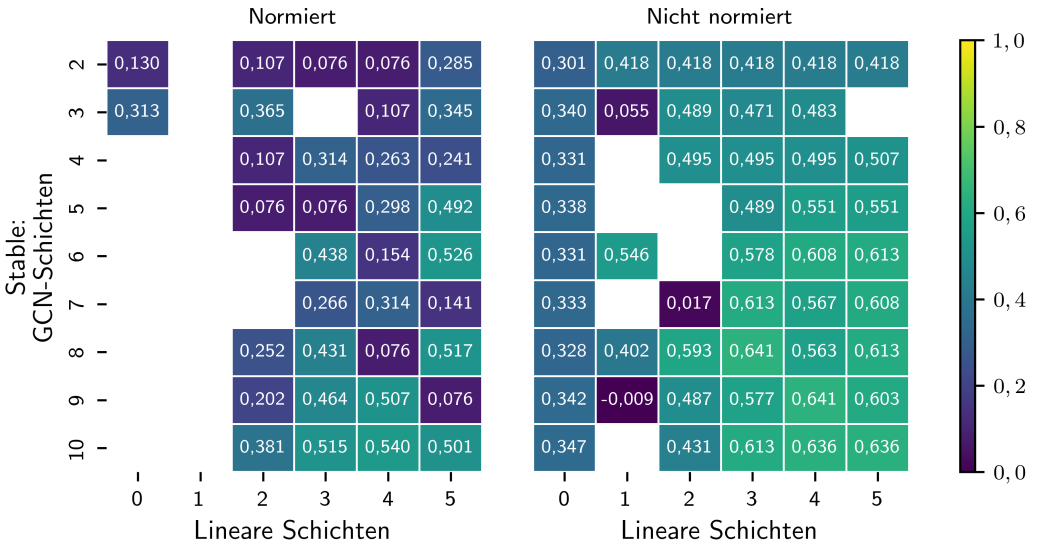

Abbildung 16: Die Mediane der MCC-Werte für den AF-Typ Stable des Experiments $E(iccma)$ sortiert nach Normierung, Anzahl der GCN-Schichten und Anzahl der linearen Schichten.

Bei den nicht normierten Designs haben die Designs mit keiner linearen Schicht niedrige MCC-Werte, die mit einer linearen Schicht sind Nein-Entscheider oder haben niedrige oder sehr niedrige Werte (bis auf unseren regelmäßigen Ausreißer `GNN6LIN1`, der mit 0,546 erneut ein besseres Ergebnis als alle normierten Designs zeigt) und die Designs mit zwei linearen Schichten zeigen gemischte Ergebnisse mit zwei Nein-Entscheidern und einem sehr niedrigen Wert in der Spitze der Keilform, sowie moderat hohen Werten bei den Designs, die nicht Teil der Keilform sind. Bei Stable-AFs werden die MCC-Werte aber schon bei recht flachen Designs stabil. Ab vier oder mehr GCN-Schichten und drei oder mehr linearen Schichten finden wir Werte von 0,489 bis 0,641. Ab sechs oder mehr GCN-Schichten erreicht die Mehrheit der Designs Werte über 0,6.

Im Vergleich zu den Werten für die Stable-AFs in $E(pbbg)$ fällt erneut die fehlende, bzw. später einsetzende Gleichförmigkeit auf und die insgesamt niedrigeren Werte.

Typ Traffic: Hier sind die normierten Designs mit keiner linearen Schicht bis auf `GCN2LIN0`, `GCN3LIN0` und `GCN4LIN0` Nein-Entscheider (siehe Abb. 17). Das `FM2`-Pendant `GCN2LIN0` selbst hat einen kaum von 0,0 verschiedenen MCC-Wert von 0,013, das `res-GCN4` ähnliche `GCN4LIN0` erreicht 0,142 und `GCN3LIN0` 0,183. Die Designs mit einer linearen Schicht erreichen nur geringe Werte unter 0,1 und diejenigen mit zwei linearen Schichten zeigen entweder niedrige Werte unter 0,15 oder moderate bis gute von 0,686 bis 0,782. Ein ähnliche Spaltung sehen wir auch bei den restlichen MCC-Werten, wobei die Anzahl der niedrigen Werte abnimmt, ohne dass heftige Ausreißer nach unten ganz verschwinden. Hohe Werte mit nur zwei Ausrei-

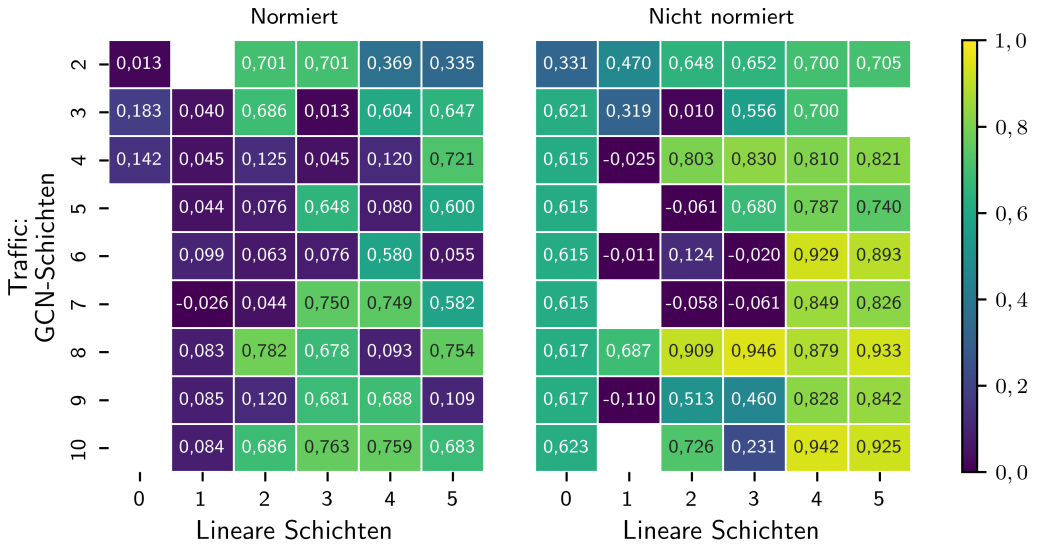

Abbildung 17: Die Mediane der MCC-Werte für den AF-Typ Traffic des Experiments $E(iccma)$ sortiert nach Normierung, Anzahl der GCN-Schichten und Anzahl der linearen Schichten.

ßern finden wir in der Region tiefer Designs mit sieben oder mehr GCN-Schichten und drei oder mehr linearen Schichten. Der höchste Wert liegt aber außerhalb des Bereichs in `GCN8LIN2` mit 0,782.

Bei den normierten Designs zeigen Designs mit keiner linearen Schicht bis auf `GNN2LIN0` mit einem MCC-Wert von 0,331 moderat hohe Werte von 0,615 bis 0,623. Diejenigen mit einer linearen Schicht zeigen gemischte Ergebnisse mit einigen Nein-Entscheider, einige schlechte Werte von -0,11 bis -0,011 und einigen moderat hohen Werten von 0,319 bis 0,687. Schlechte Werte ziehen in Keilform bis in Designs mit drei linearen Schichten hinein. Stabil gute Werte finden wir in Designs mit genau acht GCN-Schichten und zwei oder mehr linearen Schichten (0,879 bis 0,946), sowie in dem Bereich mit vier und mehr GCN-Schichten und vier oder mehr linearen Schichten (0,74 bis 0,942). Ein Muster ist auffällig: Wenn wir Ausreißer außer acht lassen, haben Designs mit einer geraden Zahl an GCN-Schichten beginnend mit vier höhere MCC-Werte als diejenigen mit ungerader Anzahl. Wir vermuten, dass dies mit der Struktur der Traffic-AFs bzw. der Translation der Verkehrsnetze in AFs zusammenhängt.

5.5.4 Zusammenfassung $E(iccma)$

Nach unserer Datensatzanalyse erwarteten wir, dass wir von den Ergebnissen von einem Teil der AF-Typen in $E(pbbg)$ auf deren Ergenisse in $E(iccma)$ schließen können. Dies hat sich z. B. für Barabási-Albert-AFs bestätigt. Von anderen Typen wuss-

ten wir, dass die AFs eines Typen in pbbg_2040 sich so sehr von AFs desselben Typen in iccma_450 unterscheiden, dass die Ergebnisse wohl nicht übertragbar sind. Und dies war der Fall für SCC-AFs. Die iccma-spezifischen AF-Typen zeigten alle interessante Ergebnisse. So sind die MCC-Werte für ABA2AF-AFs sehr gleichmäßig, für hingegen AdmBuster-AFs entweder nahe 0 oder nahe 1; Planning2AF-AFs zeigen höhere MCC-Werte auch bei normierten Designs nur für tiefe Designs; Traffic-AFs zeigen viele heftige Ausreißer nach unten und hohe MCC-Werte bei nicht normierten Designs in den Reihen mit gerader GNN-Schichtenzahl. Für die AF-Typen Erdős-Rényi, Sembuster und Watts-Strogatz, von denen kein oder fast kein AF in unserem Abschlusstestdatensatz ia Argumente enthält, wurden fast alle unsere Designs zu Nein-Entscheidern.

Nicht normierte Designs zeigen bei allen AF-Typen eine ähnlich hohe oder meistens höhere Leistungsspitze als normierte Designs. Insgesamt zeigt eine höhere summierte Schichtenanzahl (vor allem eine höhere lineare Schichtenzahl) höhere Werte, wobei im Vergleich zu $E(pbbg)$ oft Ausreißer nach unten und seltener nach oben zu finden sind. Dies steht im Gegensatz zu unserem ersten Experiment $E(pbbg)$, in dem wir große und stabile Bereiche hoher Leistung schon bei niedrigen summierten Schichtzahlen fanden. Dies liegt in der kleineren Argumentenzahl pro AF und etwas anderen Generationsparametern begründet.

Zusätzlich zum keilförmigen Bereich geringer MCC-Werte bei nicht normierten Designs fanden wir auch einen bogenförmigen Bereich geringer Werte bei bestimmten AF-Typen bei normierten Designs, für die wir keine Erklärung haben.

5.6 Experiment $E(kwt)$

Im Experiment $E(kwt)$ trainieren und testen wir Designs am nur aus kwt-AFs bestehenden kwt_2000-Datensatz und kommen zu überraschend anderen Ergebnissen.

5.6.1 Analyse der Liste:

Die MCC-Reihenfolge stimmt dieses Mal genau mit der ACC-Reihenfolge überein (vgl. Tabelle 16 auf Seite vi). Im Gegensatz zu den anderen Experimenten sind die 49 Designs mit den höchsten MCC-Werten dieses Mal allesamt normiert. Wir trennen diese ersten 49 Designs in zwei Unterbereiche. Zunächst finden wir einen großen Bereich stabil hoher MCC-Werte mit `GCN2LIN2` an Stelle 1 mit 0,93 bis `GCN4LIN3` an Stelle 45 mit 0,913. An der Spitze stehen Designs mit zwei oder drei GCN-Schichten und zwei oder mehr linearen Schichten. Danach finden wir einen kleinen Bereich rasant fallender MCC-Werte von normierten Designs ohne lineare Schichten mit `GCN8LIN0` an Stelle 46 mit einem MCC von 0,774 bis `GCN5LIN0` an Stelle 49 mit einem MCC 0,348. Schauen wir uns die Nebenkennzahlen an, so finden wir bis einschließlich Stelle 45 eine TPR von 0,997 bis meistens 1,0 mit etwas niedriger TNR von 0,921 bis 0,933. Die erfolgreichen Designs ordnen also ia perfekt oder fast perfekt zu und ina nur sehr gut. Im abfallenden Unterbereich ist im Wechsel TPR, dann TNR der höhere Wert. Die Dominanz der hohen TPR im oberen Bereich überrascht,

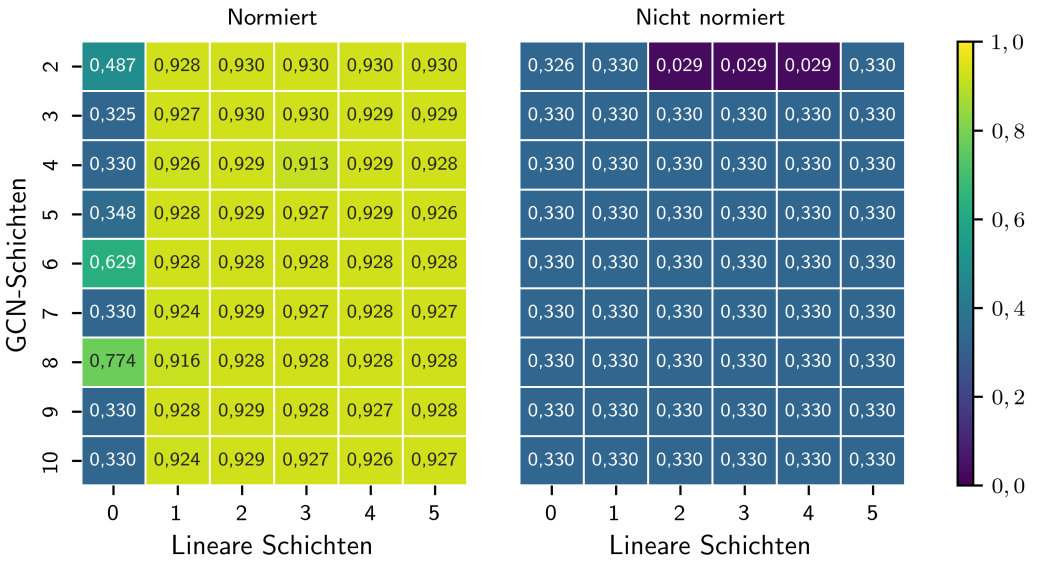

Abbildung 18: Die Mediane der MCC-Werte der verschiedenen Designs für das gesamte Experiment $E(kwt)$ sortiert nach Normierung, Anzahl der GCN-Schichten und Anzahl der linearen Schichten.[To do: Ort finden]

da die kwt-AFs insgesamt mit 54,2% zu 45.8% etwas mehr ina Argumente besitzen und somit ein KNN, dessen TPR und TNR vertauschte Werte von 0,933 und 1,0 wären, einen niedrigeren Verlust aufweisen würde als z. B. `GCN2LIN2`. Wir nehmen deshalb an, dass für kwt-AFs die korrekte Einordnung aller ia Argumente bei gleichzeitig hoher Genauigkeit der Einordnung von ina-Argumenten einfacher zu lernen ist als umgekehrt.

Die restlichen 59 Designs ordnen wir einem unteren Bereich zu. Wir finden nun zuerst 54-mal denselben MCC-Wert von 0,33 bei einer sehr niedrigen TPR von 0,186 und perfekten TNR von 1,0. Neben fast allen nicht normierten Designs nehmen auch einige wenige normierte Designs ohne lineare Schichten diesen Wert an. Das Ende des unteren Bereichs stellen `GNN2LIN0` mit 0,326, `GCN3LIN0` mit 0,325, sowie `GNN2LIN2`, `GNN2LIN3` und `GNN2LIN4` mit jeweils 0,029. Wir finden kein Design ohne validen MCC-Wert.

Die höchsten MCC-Werte mit 0,93 sind etwas höher als der von 0,927, den ein AGNN in dem ähnlichen Experiment $D_1(D_{L2}, D_{T_2})$ erreicht hat. Wir gehen davon aus, dass dieser Unterschied durch die unterschiedliche Auswahl an AFs für den Abschlusstestdatensatz zusammenhängt und mit demselben Datensätzen AGNNs und die besten GCNs die gleiche Leistung zeigen.

5.6.2 Basisanalyse nach Normierung und Schichtenzahlen:

Schauen wir uns die normierten und nicht normierten Designs nach Schichtenzahl geordnet an, so finden wir bei den normierten Designs einen großen Block sehr hoher Werte ab einer linearen Schicht(vgl. Abb. 18 auf Seite 79). Eine höhere Schichtenzahl hat einen sehr kleinen negativem Einfluss auf den MCC-Wert. Den höchsten Durchschnitt im Block haben Designs mit zwei linearen Schichten sowie solche mit zwei GCN-Schichten. Diejenigen ohne lineare Schichten zeigen gemischt viele niedrige und wenige moderat hohe MCC-Werte. Unser `FM2`-Pendant `GCN2LIN0` zeigt einen MCC von 0,487, der zwar über allen MCC nicht normierter Designs liegt, aber weit unter denen von normierten GCNs mit linearen Schichten. Dieser Wert ist auch überraschend höher als der MCC von 0,364, den `FM2` in $D_1(kwt, kwt)$ erreichte. In unserem Experiment ist die TPR mit 1,0 höher gegenüber 0,22 in D_1, die TNR niedriger mit 0,401 gegenüber 0,998, die ACC etwas höher mit 0,679 gegenüber 0,642 und die Variabilität der MCC-Werte ist mit 39,2% recht hoch. `FM2` in D_1 hat fast dieselben Werte wie `GCN5LIN0` in unserem Experiment. Wir gehen davon aus, das entweder durch Zufall oder den etwas größeren Trainingsdatensatz `GCN2LIN0` knapp in diesen besseren Werten geendet ist. Alle mit res-GCN verwandten Designs zeigen valide Werte, wobei `GCN4LIN0` und `GCN5LIN0` mit 0,33 und 0,348 ähnliche Werte wie nicht normierte Designs zeigen; nur `GCN6LIN0` findet mit einem MCC von 0,629 Anschluss an die höheren Werte der Designs mit linearen Schichten.

Die nicht normierten Designs bilden einen Block niedriger Werte ohne Ausreißer nach oben mit drei Ausreißern nach unten, die genau zwei GCN-Schichten haben. Diese Ergebnisse liegen damit stets unter dem MCC-Wert von 0,364, den `FM2` in $D_1(kwt, kwt)$ erreichte. Wir vermuten, dass den nicht normierten Designs relevante Informationen fehlen, die die normierten Designs über die transformierte Angriffsrelation, die durch die Normierung in der KIPF-WELLING-Konvolution zu einer ungerichten Relation wird, erhalten. AGNNs erhalten sie über den zweiten MLP, der dezidiert für die Nachrichtenübertragung entgegen der Angriffsrichtung zuständig ist. Selbst die durchwachsenen Ergebnisse der normierten Designs ohne lineare Schichten sind für nicht normierte nicht erreichbar. Wir prüften in einem Sonderexperiment über 4096 Läufe mit einem neuen Abschlusstestdatensatz, inwiefern das nicht normierte `GNN3LIN3` stets niedrige Werte zeigt, oder ob es eventuell doch auf eine bessere Konfiguration der Parameter stößt. Das Ergebnis entmutigt: Der MCC-Wert von 4012 der 4096 Läufe war 0,3573, ein Lauf erreichte 0,2633, ein weiterer 0,1087. Die anderen Läufe waren nahe 0,0 oder Nein-Entscheider.[109] Wir finden einen Schwellenwert vor, über den kein Modell des Designs hinauskam, und knapp 2% Ausreißer nach unten. Die Schwäche der nicht normierten Designs bei kwt-AFs steht in unseren Experimenten im Gegensatz zu allen anderen AF-Typen und so ist eine einmalige Eigenschaft.

In unseren Tests bemerken wir auch stets ein sehr frühes Konvergieren des Verlus-

[109] Ein Lauf erreichte noch einen MCC von 0,0461, 27 erreichten 0,0302, 4 erreichten 0,0301, 45 erreichten 0,03 und 5 waren Nein-Entscheider.

tes auf niedrigem Niveau. Da auch die Tiefe des Designs und damit deren Expressivität wenig Einfluss auf die Leistung haben, stellen wir die Hypothese auf, dass die Struktur von kwt-AFs für normierte Designs mit mindestens einer linearen Schicht sehr einfach zu lernen sei. Wir überprüfen dies, indem wir die versteckte Dimension verkleineren und unseren Abschlusstest mit einer wesentlich geringeren Epochenzahl durchführten. Bei einem Experiment mit `GCN2LIN3` über 128 Läufe mit 12 statt 64 versteckten Dimensionen und einer Epochenzahl von 32 statt 200 erreichen wir 73-mal den Wert 0,928 und ansonsten absteigende Werte von 0,927 bis 0,864 ohne echte Ausreißer nach unten. Dies lässt den Schluss zu, dass für kwt-AFs eine nicht sonderlich komplexe Abbildung existiert, die mit hoher (aber nicht perfekter) Genauigkeit mit den gegebenen Parametern Argumente einordnet. Diese kann durch eine höhere Expressivität des KNNs durch eine mehr versteckte Dimensionen oder einen längeren Trainingsprozess noch ein wenig von 0,928 auf 0,930 verbessert werden. Aber sie kann von nicht normierten Design nicht gefunden werden.

5.7 Zusammenfassung der Ergebnisse von E

In den drei Experimenten $E(pbbg)$, $E(iccma)$ und $E(kwt)$ haben wir gezeigt, dass sich die Leistung von GNNs über nachgeschaltete lineare Schichten stark steigern lässt. Die erreichbare Leistung unterscheidet sich stark nach dem AF-Typ und seinen Untertypen. Die drei sehr unterschiedlichen Datensätze zeigten im Detail Ergebnisse, die beim Einsatz von weniger oder gleichförmigeren Datensätzen verborgen geblieben wären. Wir haben auch gezeigt, dass `FM2` im Vergleich zu anderen GNNs, eine niedrige Leistung zeigt und weder für GCN ohne lineare Schichten noch für alle GCN-Designs exemplarische Leistung zeigt.

6 Fazit

Zunächst ist unsere Arbeit die erste, die das Problem SE_{ID} mithilfe von GNN-Techniken des maschinellen Lernens approximativ zu lösen versucht. Dies ist uns gelungen. Auch haben wir mit der Analyse der Eigenschaften unserer Datensätze und der Korrelationsmatrizen gezeigt, dass für unsere drei Datensätze pbbg_2040, iccma_450 und kwt_2000 SE_{ID}/DS_{ID} und DS_{PR} eng verwandte Probleme sind. Wir haben darüber hinaus für eine Stichprobe an Designs die Übertragbarkeit der Ergebnisse von SE_{ID} auf DS_{PR} belegt. SE_{ID} und DS_{PR} zeigten wie erwartet sehr ähnliche Ergebnisse und wir schließen daraus, dass sich SE_{ID} genauso gut (oder schlecht) wie DS_{PR} mithilfe von GNN-Techniken approximativ lösen lassen. Damit können wir unsere Ergebnisse mit denen aus anderen Forschungsarbeiten vergleichen.

Weiterhin stabilisierten wir durch eine repräsentative Auswahl des Abschlusstestdatensatzes die Ergebnisse unserer Experimente und verringerten die Standardabweichung zwischen den Medianen der MCCs verschiedener Experimente im Vergleich zur zufälligen Auswahl an AFs um fast ein Drittel.

In unserer Experimenten haben wir dann zunächst untersucht, inwiefern sich die Leistung von flachen KIPF-WELLING-GCNs wie FM2 ([KT19]) für das Problem SE_{ID} mithilfe von nachgeschalteten linearen Schichten verbessern lassen kann und haben große Verbesserungen erreicht, die die Leistungshöchstwerte für GCNs der bisherigen Forschung übertreffen.[110] Weiterhin erlauben es die linearen Schichten ähnlich wie die Graphenrohresiduen in res-GCNs ([MYNM20]) wesentlich tiefere GCNs zu nutzen, ohne dass die Modelle wie sonst üblich zu Nein-Entscheidern werden. Unsere Datensätze zeigten dabei jeweils eine andere Perspektive auf die Designs. Während sowohl pbbg_2040, das aus AFs mit bis zu 50 Argumenten generiert mit den Generatoren aus probo und AFBenchGen besteht, also auch kwt_2000, das aus kwt-AFs mit stets 151 Argumenten besteht, ab drei GCN-Schichten und einer linearen Schicht eine sehr regelmäßige Leistung für normierte Designs zeigten, fanden wir für iccma_450 kein ähnlich klares Muster. Wir fanden dort zum einen erst ab zwei linearen Schichten höhere Mediane der MCCs, zum anderen wechselten sich höhere Werte auch in diesem Bereich mit niedrigen ab. Für den AF-Typ kwt reichte die Leistung der meisten Designs sogar an die von AGGNs heran, die bisher die leistungsstärksten GNNs stellen.

Wir untersuchten dann, ob die Auslöschung der Information über die Angriffsrichtung in der Normalisierung der KIPF-WELLING-Konvolution einen negativen Einfluss auf die Leistung hat. In unseren Vergleichen zwischen normierten und nicht normierten Designs zeigten letztere für fast alle AF-Typen die höchsten Mediane der MCCs. Der Abstand zu den besten Werten der normierten Designs war zum Teil beträchtlich und die Information über die Angriffsrichtung war dazu nach unserer Vermutung nützlich. Nur bei kwt-AFs fanden wir eine niedrige, unüberbrückbare Leistungsschwelle für nicht normierte Designs. Da gerade normierte Designs und AGGNs im Einordnen der Argumente von kwt-AFs sehr erfolgreich waren, deutet dies auf ein grundsätzliches Problem von GNN-Architekturen hin, die nur Informationen in Richtung der Angriffsrelation weiterleiten. Für pbbg_2040- und iccma_450-AFs fanden wir auch einen interessanten Bereich geringer Leistung, der in den nach Schichtenart und -anzahl sortierten Tabellen wie ein Keil in die Werte hineinragt mit einem besonders tiefen Leistungseinbruch um GCN4LIN1 in pbbg_2040 und auch GCN5LIN1 in iccma_450. Für einige AF-Typen in iccma_450 fanden wir einen ähnlichen bogenförmigen Bereich bei normierten Designs. Für diese drei Phänomene haben wir keine Erklärung.

[110] Dies steht auch im Gegensatz zum Fazit von Craandijk und Bex bezüglich C_1: „In contrast, AGNN learns to perform a sequence of relational inferences which enable it to approximate the acceptance of an argument solely based on the attack structure of an AF. This more general approach **greatly outperforms** FM2's local focus." ([CB20]: S. 1672, Hervorh. d. Verf.). In unseren Tests konnte schon eine einzige angehängte lineare Schicht denn Median der MCCs von FM2 für ähnlich generierte (aber größere) AFs um mehr als 80% erhöhen. Ergänzen wir eine GCN-Schicht und noch einige lineare Schichten, dann verdoppelt er sich. Wir gehen deshalb davon aus, dass der Unterschied ‚lokaler Fokus' von FM2 gegenüber dem ‚allgemeineren Ansatz' von AGGNs höchstens einen kleinen Teil des Leistungsunterschieds verursacht. Der weit größte Teil des Leistungsunterschieds ist nach unseren Untersuchungen in den unterschiedlichen Auslesefunktionen begründet und ein weiterer Teil ist durch den Verlust der Gerichtetheit durch die Normalisierung verursacht.

Wir betrachteten auch die Ergebnisse von $E(pbbg)$ und $E(iccma)$ nach AF-Typen aufgefächert und fanden wie [KWT22] je nach Typ unterschiedlich hohe Ergebnisse. So zeigten Barabási-Albert-AFs aus beiden Datensätzen Höchstwerte von 1,0, bei Erdős-Rényi-AFs aus iccma_450 waren alle Modelle hingegen Nein-Entscheider. Dies lag bei Erdős-Rényi-AFs an der konkreten Struktur der einzelnen AFs mit nur einem AF mit wenigen ia Argumenten und bei Barabási-Albert-AFs in ihrem generell einfachen Aufbau und den daraus folgenden Eigenschaften begründet. Neben AFs vom Erdős-Rényi-Typ divergieren in ihren Eigenschaften und Ergebnissen auch die SCC- und Watts-Strogatz-AFs des pbbg_2040-Datensatzes stark von denen des iccma_450-Datensatzes, so dass wir sie unterschiedliche Untergruppen zuordnen, und erwarten, dass das Lernens mit der einen Untergruppe das Testen an der anderen nur begrenzt positiv beeinflusst.

Ein abschließendes Urteil über die Leistungsfähigkeit des einen über ein anderes Design, fällt uns schwer. Gewichten wir die Leistung für jeden unserer Datensätze pbbg_2040, iccma_450 und kwt_2000 gleich, zeigt das normierte Design `GCN8LIN5` die höchste Summe der Mediane der MCCs mit Werten von 0,786, 0,776 und 0,928. Das nicht normierte Design mit der höchsten Summe ist `GNN10LIN4`[111] und hat Werte des Median der MCCs von 0,878 (+0,098), 0,903 (+0,127) und 0,330 (-0,598). Gewichten wir jedoch die Werte nach Anzahl und Anteil an jedem AF-Typ[112], so liegt `GNN10LIN4` mit 0,849 vor `GCN8LIN5` mit 0,791. Eine weitere Möglichkeit wäre es, die drei Abschlusstestdatensätze zusammenzufassen, und den MCC eines virtuellen Gesamtexperimentes zu berechnen. Dann würde die Anzahl an Argumenten pro AF-Typ einen großen Anteil am Endergebnis haben und der kwt-Typ mit seinen 271.800 Argumenten würde fast die Hälfte der Argumente des Abschlusstestdatensatzes stellen. Wir könnten auch die Leistung zusätzlich über die Rechenzeit gewichten (immerhin ist der Echtzeiteinsatz eine Hauptmotivation für das approximative Verfahren). Für normierte Designs verdoppelt die Berechnung und Anwendung der normierten Laplace-Matrix knapp die Laufzeit einer GNN-Schicht im Vergleich zu einer nicht normierten, weshalb z. B. `GNN10LIN4` und `GCN5LIN4` ähnliche Laufzeiten haben. Weiterhin kosten fünf lineare Schichten weniger Zeit als eine weitere GNN-Schicht. Folglich beruht jede Aussage, welches Design das leistungsstärkste[113] Design ist, auf einer Entscheidung zur Gewichtung der Ergebnisse, die wir hier nicht treffen möchten.

Für weiterführende Forschungvorhaben bietet sich eine Reihe von verschiedenen Ansätzen an. So könnte unsere Vermutung für die starke Leistung nicht-normierter Designs bei den meisten AF-Typen und für die schwache Leistung bei kwt-AFs dadurch überprüft werden, dass zusätzlich zu KIPF-WELLING-normierten GCNs und

[111] `GNN8LIN3` zeigt dieselbe gerundete Summe, fällt aber knapp hinter `GNN10LIN4` zurück, wenn wir die ungerundeten Rohdaten gebrauchen.

[112] Wir nutzen diese einfache Gewichtung: $(MCC(pbbg) \times (6/2) + MCC(iccma) \times (5 + (6/2)) + MCC(kwt))/12$.

[113] Wir dabei auch schon die Entscheidung getroffen, dass der Median der MCCs aus fünf Läufen das zentrale Maß der Leistungstärke ist.

nicht normierten GNNs noch ein weiteres, heterogenes GNN[114] herangezogen wird, das wie ein AGNN die Informationen entlang der und gegen die Angriffsrelation unabhängig voneinander sendet. Designs dieser drei Architekturen sollten an kwt- und anderen AF-Typen getestet werden, um die Leistungsgrenzen von GCN-Architekturen nachzuziehen. Danach sollten die leistungsstärksten Designs mit einem AGGN bezüglich AFs unterschiedlicher Größe verglichen werden, um inbetracht der von uns produzierten Erkenntnisse eine Neubewertung der Stärken und Schwächen von AGGN- und GCN-Architekturen zu ermöglichen.

Eine tiefergehende Untersuchung sind ebenfalls die Muster wert, die in den nach Schichtenart und -anzahl sortierten Tabellen auftreten. Allen voran die Bereiche niedriger Leistung wie die Keilform in nicht normierten Designs oder die Bögen in normierten Designs und deren mögliche Ursachen, sowie die Bereiche stabil hoher Werte mit tiefen Designs und deren Ausdehnung jenseits von 10 GCN-Schichten oder 5 linearen Schichten. Daran schließt die Frage an, ob und wie wie GCN- und GNN-Designs mit weiteren GCN- und linearen Schichten skalieren, oder ob die Leistung ab einer bestimmten Anzahl rapide abfällt oder jenseits eines bestimmten Wertes ansteigt.

In unserer Arbeit haben wir lineare Schichten nur auf GCN-Schichten folgen lassen. Damit sind hybride GNNs, die normierte und/oder nicht normierte Schichten mit dazwischen liegenden und/oder nachfolgenden linearen Schichten beinhalten, noch unerforscht. So könnte die Leistung von z. B. `GNN6LIN5` mit den auch 11-schichtigen Hybriden `GNN3LIN2GCN3LIN3` oder `GCN3LIN1GNN3LIN4` verglichen werden, die vielleicht die Stärken (oder Schwächen) der normierten, nicht normierten und linearen Schichten neu kombinieren.

Und schließlich deuten die unterschiedlichen Eigenschaften und Ergebnisse der Untergruppen einiger AF-Typen auf erweiterte oder alternative, aber für das Trainieren und Testen unserer GNNs sinnvolle AF-Klassifizierungen hin, die z. B. mit einem auf Graphklassifikation spezialisierten GNN untersucht werden können.

[114] Dies kann über eine bestehende GNN-Architektur wie r-GCN in [SKB+17] oder eine eigene Implementation über z. B. `torch_geometric.nn.conv.HeteroConv` realisiert werden.

Literatur

[BKI19] Christoph Beierle und Gabriele Kern-Isberner. *Methoden wissensbasierter Systeme: Grundlagen, Algorithmen, Anwendungen*. Computational Intelligence. Springer, 6., überarbeitete Auflage, 2019.

[CB20] Dennis Craandijk und Floris Bex. Deep learning for abstract argumentation semantics. In Christian Bessiere (Hrsg.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, S. 1667–1673. International Joint Conferences on Artificial Intelligence Organization, 2020.

[CB22] Dennis Craandijk und Floris Bex. Enforcement heuristics for argumentation with deep reinforcement learning. In Matthijs Spaan Katia Sycara, Vasant Honavar (Hrsg.), *Thirty-Sixth AAAI Conference on Artificial Intelligence Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence The Twelveth Symposium on Educational Advances in Artificial Intelligence*, Proceedings of the AAAI Conference on Artificial Intelligence (Band 36.5), S. 5573–5581. AAAI Press, 2022.

[CGV] Federico Cerutti, Massimiliano Giacomin und Mauro Vallati. Exploiting planning problems for generating challenging abstract argumentation frameworks. `https://argumentationcompetition.org/2017/Planning2AF.pdf`. (Zuletzt abgerufen am 11.1.2023).

[COS+14] Federico Cerutti, Nir Oren, Hannes Strass, Matthias Thimm und Mauro Vallati. A benchmark framework for a computational argumentation competition. In Simon Parsons, Nir Oren, Chris Reed und Federico Cerutti (Hrsg.), *Computational Models of Argument*, Frontiers in Artificial Intelligence and Applications (Band 266), S. 459–460. IOS Press, 2014.

[CP] Martin Caminada und Mikołaj Podlaszewski. A benchmark example for (strong) admissibility. `https://argumentationcompetition.org/2017/AdmBuster.pdf`. (Zuletzt abgerufen am 11.1.2023).

[CV] Martin Caminada und Bart Verheij. A benchmark example for semi-stable semantics. `http://argumentationcompetition.org/2017/SemBuster.pdf`. (Zuletzt abgerufen am 11.1.2023).

[CWH+20] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding und Yaliang Li. Simple and deep graph convolutional networks. In Hal Daumé III und Aarti Singh (Hrsg.), *Proceedings of the 37th International Conference on Machine Learning*, Proceedings of Machine Learning Research (Band 119), S. 1725–1735. PMLR, 2020.

[DBC02] Paul E. Dunne und T.J.M. Bench-Capon. Coherence in finite argument systems. *Artificial Intelligence*, 141(1):187–203, 2002.

[Dil] Martin Diller. Traffic networks become argumentation frameworks. `https://argumentationcompetition.org/2017/Traffic.pdf`. (Zuletzt abgerufen am 11.1.2023).

[DMT07] Phan Minh Dung, Paolo Mancarella und Francesca Toni. Computing ideal sceptical argumentation. *Artificial Intelligence*, 171(10-15):642–674, 2007.

[Dun95] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–358, 1995.

[Dun09] Paul E. Dunne. The computational complexity of ideal semantics. *Artificial Intelligence*, 173(18):1559–1591, 2009.

[GBC16] Ian Goodfellow, Yoshua Bengio und Aaron Courville. *Deep Learning*. MIT Press, 2016.

[GCV14] Massimiliano Giacomin, Federico Cerutti und Mauro Vallati. Generating challenging benchmark AFs. In Simon Parsons, Nir Oren, Chris Reed und Federico Cerutti (Hrsg.), *Computational Models of Argument*, Frontiers in Artificial Intelligence and Applications (Band 266), S. 457–458. IOS Press, 2014.

[GLMW] Sarah Alice Gaggl, Thomas Linsbichler, Marco Maratea und Stefan Woltran. Benchmark selection at ICCMA'17. `http://argumentationcompetition.org/2017/benchmark_selection_iccma2017.pdf`. (Zuletzt abgerufen am 11.1.2023).

[HSW89] Kurt Hornik, Maxwell Stinchcombe und Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[HVG11] David K. Hammond, Pierre Vandergheynst und Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[KL20] Patrick Kidger und Terry Lyons. Universal approximation with deep narrow networks. In Jacob Abernethy und Shivani Agarwal (Hrsg.), *Conference on Learning Theory*, Proceedings of Machine Learning Research (Band 125), S. 2306–2327. PMLR, 2020.

[KT19] Isabelle Kuhlmann und Matthias Thimm. Using graph convolutional networks for approximate reasoning with abstract argumentation frameworks: A feasibility study. In Nahla Ben Amor, Benjamin Quost und Martin Theobald (Hrsg.), *Proceedings of the 13th International Conference on Scalable Uncertainty Management)*, Lecture Notes in Computer Science (Band 11940), S. 24–37. Springer, 2019.

[KW17] Thomas N. Kipf und Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations*, 2017. arXiv preprint arXiv:1609.02907.

[KWT22] Isabelle Kuhlmann, Thorsten Wujek und Matthias Thimm. On the impact of data selection when applying machine learning in abstract argumentation. In Francesca Toni, Sylwia Polberg, Richard Booth, Martin Caminada und Hiroyuki Kido (Hrsg.), *Computational Models of Argument*, Frontiers in Artificial Intelligence and Applications (Band 353), S. 224–235. IOS Press, 2022.

[LHW18] Qimai Li, Zhichao Han und Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In Kilian Q. Weinberger Sheila A. McIlraith (Hrsg.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, Proceedings of the AAAI Conference on Artificial Intelligence (Band 32.1), S. 3538–3545. AAAI Press, 2018.

[LWJ] Tuomo Lehtonen, Johannes P. Wallner und Matti Järvisalo. Assumption-based argumentation translated to argumentation frameworks. `https://argumentationcompetition.org/2017/ABA2AF.pdf`. (Zuletzt abgerufen am 11.1.2023).

[Mal08] Stephane Mallat. *A Wavelet Tour of Signal Processing: The Sparse Way (3. erweiterte Ausgabe)*. Elsevier, 2008.

[Mit97] Tom M. Mitchell. *Machine learning*. McGraw-Hill, 1997.

[MP43] Warren Sturgis McCulloch und Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.

[MP69] Marvin Minsky und Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969.

[MYNM20] Lars Malmqvist, Tommy Yuan, Peter Nightingale und Suresh Manandhar. Determining the acceptability of abstract arguments with graph convolutional networks. In Sarah Alice Gaggl, Matthias Thimm und

Mauro Vallati (Hrsg.), *Proceedings of the Third International Workshop on Systems and Algorithms for Formal Argumentation co-located with the 8th International Conference on Computational Models of Argument*, CEUR Workshop Proceedings (Band 2672), S. 47–56. CEUR-WS.org, 2020.

[NJ20] Andreas Niskanen und Matti Järvisalo. μ-toksia: An efficient abstract argumentation reasoner. In Diego Calvanese, Esra Erdem und Michael Thielscher (Hrsg.), *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning*, S. 800–804, 2020.

[OS19] Kenta Oono und Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification, 2019. arXiv preprint arXiv:1905.10947.

[PARS14] Bryan Perozzi, Rami Al-Rfou und Steven Skiena. DeepWalk: Online learning of social representations. In Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang und Rayid Ghani (Hrsg.), *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, S. 701–710. Association for Computing Machinery, 2014.

[PHvIP22] Yulong Pei, Tianjin Huang, Werner van Ipenburg und Mykola Pechenizkiy. Resgcn: attention-based deep residual modeling for anomaly detection on attributed networks. *Machine Learning*, 111(2):519–541, 2022.

[RHW86] David E. Rumelhart, Geoffrey E. Hinton und Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[Ros58] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever und Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[SKB+17] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov und Max Welling. Modeling relational data with graph convolutional networks. In Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai und Mehwish Alam (Hrsg.), *The Semantic Web. ESWC 2018*, Lecture Notes in Computer Science (Band 10843), S. 593–607. Springer, 2017.

[YYL20] Jiaxuan You, Rex Ying und Jure Leskovec. Design space for graph neural networks. In *NIPS'20: Proceedings of the 34th International Conference on Neural Information Processing Systems*, S. 17009–17021. Curran Associates Inc., 2020.

[ZCH+20] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li und Maosong Sun. Graph Neural Networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[ZM19] Jiawei Zhang und Lin Meng. GResNet: Graph Residual Network for reviving deep GNNs from suspended animation, 2019. arXiv preprint arXiv:1909.05729.

[ZRR+21] Jiong Zhu, Ryan A. Rossi, Anup Rao, Tung Mai, Nedim Lipka, Nesreen K. Ahmed und Danai Koutra. Graph neural networks with heterophily. In *Thirty-Fifth AAAI Conference on Artificial Intelligence and Thirty-Third Conference on Innovative Applications of Artificial Intelligence and Eleventh Symposium on Educational Advances in Artificial Intelligence*, Proceedings of the AAAI Conference on Artificial Intelligence (Band 35.12), S. 11168–11176. AAAI Press, 2021.

Anhang

7 Ergebnislisten der Experimente

Es folgen die Ergebnislisten der 5 Läufe pro Design der Experimente E. Die TPR, TNR und ACC gehören zu dem Lauf, der den Median der MCC-Werte stellt.

Tabelle 14: Ergebnisliste von $E(pbbg)$.

Architektur	GCN-Schichten	Lineare Schichten	Normalisiert	MCC: Median	MCC: CV	TPR	TNR	ACC
GNN8LIN3	8	3	Nein	0,891	0,4%	0,831	1,0	0,964
GNN9LIN4	9	4	Nein	0,887	1,6%	0,824	1,0	0,962
GNN8LIN5	8	5	Nein	0,885	0,4%	0,821	1,0	0,962
GNN9LIN5	9	5	Nein	0,885	0,7%	0,821	1,0	0,962
GNN7LIN5	7	5	Nein	0,884	0,4%	0,82	1,0	0,961
GNN9LIN3	9	3	Nein	0,882	0,4%	0,818	1,0	0,961
GNN10LIN2	10	2	Nein	0,879	0,6%	0,813	1,0	0,96
GNN5LIN5	5	5	Nein	0,878	0,3%	0,819	0,998	0,96
GNN10LIN5	10	5	Nein	0,878	0,8%	0,811	1,0	0,959
GNN7LIN4	7	4	Nein	0,878	0,1%	0,811	1,0	0,959
GNN8LIN4	8	4	Nein	0,878	1,6%	0,811	1,0	0,959
GNN10LIN3	10	3	Nein	0,878	1,4%	0,811	1,0	0,959
GNN7LIN3	7	3	Nein	0,878	0,5%	0,811	1,0	0,959
GNN10LIN4	10	4	Nein	0,878	1,7%	0,811	1,0	0,959
GNN7LIN2	7	2	Nein	0,878	43,2%	0,811	1,0	0,959
GNN6LIN5	6	5	Nein	0,878	0,2%	0,81	1,0	0,959
GNN6LIN4	6	4	Nein	0,878	0,2%	0,815	0,999	0,959
GNN6LIN3	6	3	Nein	0,877	0,8%	0,814	0,999	0,959
GNN5LIN4	5	4	Nein	0,876	0,6%	0,808	1,0	0,959
GNN5LIN3	5	3	Nein	0,872	2,0%	0,805	0,999	0,958
GNN9LIN1	9	1	Nein	0,866	7,6%	0,862	0,981	0,956
GNN4LIN5	4	5	Nein	0,861	0,5%	0,823	0,99	0,954
GNN4LIN4	4	4	Nein	0,857	0,4%	0,796	0,996	0,953
GNN4LIN3	4	3	Nein	0,852	3,8%	0,805	0,992	0,952
GNN5LIN2	5	2	Nein	0,849	29,6%	0,772	0,999	0,95
GNN8LIN2	8	2	Nein	0,839	1,8%	0,758	0,998	0,947
GNN3LIN3	3	3	Nein	0,835	0,2%	0,756	0,998	0,946
GNN4LIN2	4	2	Nein	0,835	5,6%	0,746	1,0	0,946
GNN3LIN4	3	4	Nein	0,834	0,2%	0,765	0,995	0,946
GNN3LIN5	3	5	Nein	0,832	0,8%	0,749	0,998	0,945
GNN9LIN2	9	2	Nein	0,82	6,5%	0,754	0,992	0,941
GCN3LIN5	3	5	Ja	0,82	1,1%	0,738	0,997	0,941
GCN3LIN3	3	3	Ja	0,819	0,7%	0,728	0,999	0,941
GCN3LIN4	3	4	Ja	0,816	4,0%	0,723	0,999	0,94
GCN3LIN2	3	2	Ja	0,813	1,4%	0,721	0,998	0,939
GCN4LIN2	4	2	Ja	0,807	0,2%	0,716	0,997	0,937
GCN4LIN5	4	5	Ja	0,805	0,4%	0,723	0,995	0,937
GCN4LIN4	4	4	Ja	0,804	0,9%	0,722	0,995	0,936
GCN4LIN3	4	3	Ja	0,804	0,3%	0,715	0,996	0,936
GCN5LIN2	5	2	Ja	0,803	0,2%	0,723	0,994	0,936
GCN5LIN5	5	5	Ja	0,802	0,4%	0,72	0,994	0,936
GCN6LIN3	6	3	Ja	0,802	1,4%	0,715	0,995	0,935
GCN5LIN4	5	4	Ja	0,801	0,4%	0,715	0,995	0,935

Fortsetzung auf der nächsten Seite

Tabelle 14 – Fortsetzung von der letzten Seite

Architektur	GCN-Schichten	Lineare Schichten	Normalisiert	MCC: Median	MCC: CV	TPR	TNR	ACC
GCN5LIN3	5	3	Ja	0,8	0,5%	0,726	0,992	0,935
GCN6LIN4	6	4	Ja	0,8	0,2%	0,718	0,994	0,935
GCN7LIN2	7	2	Ja	0,8	0,2%	0,716	0,994	0,935
GCN4LIN1	4	1	Ja	0,8	0,3%	0,762	0,982	0,935
GCN6LIN2	6	2	Ja	0,799	0,2%	0,718	0,994	0,935
GCN6LIN5	6	5	Ja	0,798	0,2%	0,726	0,991	0,935
GCN5LIN1	5	1	Ja	0,798	0,3%	0,766	0,98	0,935
GCN8LIN2	8	2	Ja	0,798	0,4%	0,719	0,993	0,934
GCN9LIN2	9	2	Ja	0,796	0,8%	0,718	0,992	0,934
GCN7LIN4	7	4	Ja	0,795	0,3%	0,726	0,99	0,934
GCN7LIN3	7	3	Ja	0,795	0,1%	0,727	0,99	0,934
GCN10LIN2	10	2	Ja	0,795	0,5%	0,726	0,99	0,934
GCN7LIN1	7	1	Ja	0,793	0,7%	0,761	0,98	0,933
GCN9LIN3	9	3	Ja	0,792	0,8%	0,738	0,986	0,933
GCN8LIN5	8	5	Ja	0,792	0,2%	0,721	0,99	0,933
GCN8LIN3	8	3	Ja	0,791	0,4%	0,729	0,988	0,932
GNN10LIN1	10	1	Nein	0,79	9,6%	0,799	0,967	0,931
GCN10LIN5	10	5	Ja	0,788	2,0%	0,742	0,983	0,931
GCN8LIN1	8	1	Ja	0,787	0,4%	0,773	0,974	0,931
GCN10LIN4	10	4	Ja	0,786	0,2%	0,746	0,981	0,931
GCN9LIN5	9	5	Ja	0,786	6,0%	0,739	0,983	0,931
GCN8LIN4	8	4	Ja	0,785	1,0%	0,715	0,989	0,931
GCN10LIN1	10	1	Ja	0,785	0,3%	0,76	0,977	0,93
GCN9LIN4	9	4	Ja	0,785	0,5%	0,74	0,982	0,931
GCN9LIN1	9	1	Ja	0,783	0,2%	0,746	0,98	0,93
GCN10LIN3	10	3	Ja	0,783	1,4%	0,754	0,978	0,93
GCN3LIN1	3	1	Ja	0,779	1,2%	0,771	0,971	0,928
GCN6LIN1	6	1	Ja	0,779	1,5%	0,761	0,974	0,928
GNN3LIN2	3	2	Nein	0,774	5,2%	0,675	0,995	0,927
GCN7LIN5	7	5	Ja	0,77	5,9%	0,735	0,978	0,926
GCN2LIN1	2	1	Ja	0,733	0,2%	0,696	0,974	0,915
GCN2LIN4	2	4	Ja	0,728	2,3%	0,703	0,97	0,913
GCN2LIN5	2	5	Ja	0,728	1,6%	0,764	0,95	0,91
GNN2LIN5	2	5	Nein	0,722	0,6%	0,667	0,978	0,912
GNN2LIN4	2	4	Nein	0,721	0,4%	0,674	0,976	0,911
GNN2LIN3	2	3	Nein	0,72	0,4%	0,657	0,98	0,911
GCN2LIN3	2	3	Ja	0,719	0,8%	0,677	0,974	0,911
GCN2LIN2	2	2	Ja	0,71	1,7%	0,631	0,984	0,908
GNN2LIN1	2	1	Nein	0,683	9,0%	0,682	0,957	0,898
GNN2LIN2	2	2	Nein	0,679	3,5%	0,593	0,983	0,899
GCN5LIN0	5	0	Ja	0,594	0,0%	0,41	1,0	0,874
GCN4LIN0	4	0	Ja	0,588	0,9%	0,41	0,998	0,872
GNN6LIN2	6	2	Nein	0,581	20,5%	0,397	0,999	0,87
GNN8LIN1	8	1	Nein	0,571	33,6%	0,403	0,995	0,868
GNN3LIN1	3	1	Nein	0,559	41,2%	0,891	0,768	0,795
GCN3LIN0	3	0	Ja	0,557	0,4%	0,41	0,99	0,866
GNN8LIN0	8	0	Nein	0,556	0,2%	0,878	0,775	0,797
GNN6LIN0	6	0	Nein	0,556	0,3%	0,878	0,775	0,797
GNN5LIN0	5	0	Nein	0,556	0,4%	0,878	0,775	0,797
GNN7LIN0	7	0	Nein	0,555	4,8%	0,878	0,774	0,796

Fortsetzung auf der nächsten Seite

Tabelle 14 – Fortsetzung von der letzten Seite

Architektur	GCN-Schichten	Lineare Schichten	Normalisiert	MCC: Median	MCC: CV	TPR	TNR	ACC
GNN9LIN0	9	0	Nein	0,553	10,9%	0,878	0,772	0,795
GNN4LIN0	4	0	Nein	0,478	0,0%	0,485	0,936	0,839
GNN10LIN0	10	0	Nein	0,478	0,0%	0,485	0,936	0,839
GNN3LIN0	3	0	Nein	0,477	0,0%	0,487	0,935	0,839
GNN7LIN1	7	1	Nein	0,477	30,6%	0,321	0,989	0,846
GNN2LIN0	2	0	Nein	0,473	0,0%	0,554	0,903	0,828
GNN6LIN1	6	1	Nein	0,417	33,4%	0,266	0,988	0,833
GCN2LIN0	2	0	Ja	0,408	0,2%	0,612	0,828	0,782
GNN5LIN1	5	1	Nein	0,301	57,7%	0,146	0,993	0,811
GNN4LIN1	4	1	Nein	0,197	59,3%	0,106	0,983	0,796
GCN6LIN0	6	0	Ja	– (Nein)	0,0%	0,0	1,0	0,786
GCN7LIN0	7	0	Ja	– (Nein)	0,0%	0,0	1,0	0,786
GCN8LIN0	8	0	Ja	– (Nein)	0,0%	0,0	1,0	0,786
GCN9LIN0	9	0	Ja	– (Nein)	0,0%	0,0	1,0	0,786
GCN10LIN0	10	0	Ja	– (Nein)	0,0%	0,0	1,0	0,786

Ende von Tabelle 14

Tabelle 15: Ergebnisliste von $E(iccma)$.

Architektur	GCN-Schichten	Lineare Schichten	Normalisiert	MCC: Median	MCC: CV	TPR	TNR	ACC
GNN10LIN4	10	4	Nein	0,903	0,0%	0,848	0,998	0,984
GNN8LIN3	8	3	Nein	0,89	11,0%	0,831	0,998	0,982
GNN8LIN5	8	5	Nein	0,89	5,3%	0,812	1,0	0,982
GNN10LIN5	10	5	Nein	0,881	17,5%	0,829	0,996	0,98
GNN6LIN4	6	4	Nein	0,878	1,1%	0,796	0,999	0,98
GNN8LIN2	8	2	Nein	0,862	1,3%	0,776	0,999	0,978
GNN9LIN5	9	5	Nein	0,848	7,3%	0,812	0,992	0,975
GNN7LIN5	7	5	Nein	0,833	3,9%	0,807	0,99	0,973
GNN6LIN5	6	5	Nein	0,832	28,9%	0,864	0,982	0,971
GNN8LIN4	8	4	Nein	0,824	21,7%	0,755	0,994	0,972
GNN9LIN4	9	4	Nein	0,817	4,5%	0,759	0,992	0,97
GNN4LIN5	4	5	Nein	0,792	4,1%	0,773	0,986	0,966
GNN5LIN4	5	4	Nein	0,791	21,9%	0,759	0,987	0,966
GCN10LIN4	10	4	Ja	0,776	9,6%	0,754	0,985	0,963
GNN4LIN3	4	3	Nein	0,769	21,8%	0,631	0,998	0,964
GCN8LIN5	8	5	Ja	0,761	6,1%	0,707	0,988	0,962
GNN4LIN4	4	4	Nein	0,758	2,2%	0,611	0,999	0,962
GCN10LIN3	10	3	Ja	0,756	14,7%	0,666	0,992	0,962
GNN5LIN5	5	5	Nein	0,745	8,0%	0,73	0,982	0,958
GCN10LIN2	10	2	Ja	0,734	14,5%	0,673	0,988	0,958
GCN7LIN3	7	3	Ja	0,721	18,9%	0,671	0,985	0,956
GCN9LIN3	9	3	Ja	0,718	48,6%	0,686	0,983	0,955
GNN7LIN4	7	4	Nein	0,715	8,4%	0,537	1,0	0,956
GNN3LIN4	3	4	Nein	0,713	0,0%	0,672	0,983	0,954
GNN5LIN3	5	3	Nein	0,699	27,3%	0,683	0,979	0,951
GCN8LIN3	8	3	Ja	0,697	25,8%	0,597	0,991	0,953
GNN2LIN5	2	5	Nein	0,696	4,8%	0,521	0,999	0,954

Fortsetzung auf der nächsten Seite

Tabelle 15 – Fortsetzung von der letzten Seite

Architektur	GCN-Schichten	Lineare Schichten	Normalisiert	MCC: Median	MCC: CV	TPR	TNR	ACC
GNN8LIN1	8	1	Nein	0,695	3,0%	0,645	0,984	0,952
GNN2LIN4	2	4	Nein	0,694	2,2%	0,544	0,996	0,954
GCN5LIN3	5	3	Ja	0,69	20,0%	0,562	0,994	0,953
GCN4LIN5	4	5	Ja	0,676	14,9%	0,52	0,996	0,951
GNN2LIN3	2	3	Nein	0,673	40,0%	0,521	0,996	0,951
GNN2LIN2	2	2	Nein	0,663	3,1%	0,545	0,991	0,949
GCN3LIN2	3	2	Ja	0,66	19,4%	0,489	0,997	0,949
GCN8LIN2	8	2	Ja	0,659	13,9%	0,688	0,969	0,942
GCN10LIN5	10	5	Ja	0,653	17,1%	0,769	0,95	0,933
GCN3LIN5	3	5	Ja	0,647	6,0%	0,519	0,992	0,947
GCN5LIN5	5	5	Ja	0,628	26,6%	0,816	0,93	0,919
GNN9LIN2	9	2	Nein	0,613	28,2%	0,763	0,938	0,921
GNN3LIN3	3	3	Nein	0,61	3,5%	0,599	0,972	0,937
GNN10LIN2	10	2	Nein	0,609	24,0%	0,556	0,979	0,939
GCN6LIN4	6	4	Ja	0,591	30,3%	0,612	0,965	0,932
GNN9LIN3	9	3	Nein	0,568	30,4%	0,859	0,889	0,886
GCN9LIN2	9	2	Ja	0,567	21,2%	0,386	0,996	0,938
GCN3LIN4	3	4	Ja	0,565	40,5%	0,581	0,964	0,928
GCN7LIN4	7	4	Ja	0,564	18,3%	0,411	0,992	0,937
GCN9LIN4	9	4	Ja	0,544	42,0%	0,463	0,981	0,932
GNN2LIN1	2	1	Nein	0,542	8,2%	0,548	0,965	0,926
GCN7LIN5	7	5	Ja	0,535	22,7%	0,631	0,943	0,914
GCN6LIN3	6	3	Ja	0,534	29,3%	0,367	0,994	0,934
GCN6LIN2	6	2	Ja	0,511	10,3%	0,367	0,99	0,931
GCN5LIN2	5	2	Ja	0,508	26,5%	0,366	0,99	0,931
GNN10LIN3	10	3	Nein	0,486	28,4%	0,268	0,999	0,93
GCN2LIN2	2	2	Ja	0,485	13,6%	0,254	1,0	0,93
GCN5LIN4	5	4	Ja	0,474	37,2%	0,359	0,985	0,926
GCN4LIN2	4	2	Ja	0,467	57,8%	0,284	0,995	0,928
GCN2LIN3	2	3	Ja	0,466	25,7%	0,242	0,999	0,928
GCN2LIN4	2	4	Ja	0,465	23,8%	0,5	0,954	0,911
GNN6LIN2	6	2	Nein	0,457	4,2%	0,258	0,997	0,927
GCN4LIN3	4	3	Ja	0,455	36,5%	0,301	0,991	0,926
GNN6LIN1	6	1	Nein	0,454	32,8%	0,228	1,0	0,927
GNN10LIN0	10	0	Nein	0,448	0,0%	0,323	0,986	0,924
GCN2LIN5	2	5	Ja	0,446	7,3%	0,545	0,934	0,898
GNN3LIN0	3	0	Nein	0,445	0,0%	0,32	0,987	0,924
GNN6LIN3	6	3	Nein	0,444	31,5%	0,228	0,999	0,926
GNN9LIN0	9	0	Nein	0,442	0,0%	0,325	0,985	0,923
GNN8LIN0	8	0	Nein	0,44	15,5%	0,333	0,984	0,922
GNN6LIN0	6	0	Nein	0,44	0,1%	0,33	0,984	0,922
GNN5LIN0	5	0	Nein	0,439	0,0%	0,327	0,985	0,922
GNN4LIN0	4	0	Nein	0,439	0,0%	0,328	0,984	0,922
GNN7LIN0	7	0	Nein	0,439	0,0%	0,327	0,984	0,922
GNN4LIN1	4	1	Nein	0,433	0,6%	0,229	0,997	0,925
GNN4LIN2	4	2	Nein	0,422	32,4%	0,229	0,996	0,924
GCN3LIN1	3	1	Ja	0,412	38,2%	0,286	0,987	0,921
GCN10LIN1	10	1	Ja	0,406	3,4%	0,382	0,966	0,911
GCN9LIN1	9	1	Ja	0,405	1,6%	0,369	0,969	0,912
GCN8LIN1	8	1	Ja	0,402	11,9%	0,374	0,967	0,911

Fortsetzung auf der nächsten Seite

Tabelle 15 – Fortsetzung von der letzten Seite

Architektur	GCN-Schichten	Lineare Schichten	Normalisiert	MCC: Median	MCC: CV	TPR	TNR	ACC
GCN8LIN4	8	4	Ja	0,399	54,7%	0,357	0,97	0,912
GCN7LIN2	7	2	Ja	0,396	53,4%	0,382	0,963	0,908
GNN3LIN2	3	2	Nein	0,392	30,2%	0,262	0,988	0,919
GCN6LIN5	6	5	Ja	0,392	29,6%	0,353	0,969	0,911
GCN7LIN1	7	1	Ja	0,377	25,5%	0,296	0,978	0,914
GNN3LIN1	3	1	Nein	0,372	67,5%	0,82	0,761	0,766
GCN4LIN4	4	4	Ja	0,364	43,1%	0,342	0,964	0,906
GNN5LIN2	5	2	Nein	0,355	1,8%	0,228	0,988	0,916
GNN7LIN3	7	3	Nein	0,352	41,9%	0,228	0,987	0,916
GNN7LIN2	7	2	Nein	0,349	3,9%	0,229	0,987	0,915
GCN5LIN1	5	1	Ja	0,333	71,7%	0,282	0,972	0,907
GCN4LIN1	4	1	Ja	0,31	31,6%	0,289	0,963	0,9
GCN3LIN3	3	3	Ja	0,31	19,1%	0,285	0,965	0,901
GCN9LIN5	9	5	Ja	0,306	12,9%	0,157	0,993	0,914
GCN3LIN0	3	0	Ja	0,286	4,7%	0,725	0,729	0,728
GCN2LIN0	2	0	Ja	0,247	0,1%	0,879	0,544	0,576
GNN9LIN1	9	1	Nein	0,222	40,3%	0,228	0,955	0,887
GNN2LIN0	2	0	Nein	0,218	0,5%	0,424	0,857	0,816
GCN4LIN0	4	0	Ja	0,216	0,0%	0,051	1,0	0,911
GCN6LIN1	6	1	Ja	0,187	58,0%	0,068	0,996	0,908
GNN7LIN1	7	1	Nein	-0,002	0,0%	0,0	1,0	0,906
GCN2LIN1	2	1	Ja	– (Nein)	0,0%	0,0	1,0	0,906
GCN5LIN0	5	0	Ja	– (Nein)	0,0%	0,0	1,0	0,906
GCN6LIN0	6	0	Ja	– (Nein)	0,0%	0,0	1,0	0,906
GCN7LIN0	7	0	Ja	– (Nein)	0,0%	0,0	1,0	0,906
GCN8LIN0	8	0	Ja	– (Nein)	0,0%	0,0	1,0	0,906
GCN9LIN0	9	0	Ja	– (Nein)	0,0%	0,0	1,0	0,906
GCN10LIN0	10	0	Ja	– (Nein)	0,0%	0,0	1,0	0,906
GNN3LIN5	3	5	Nein	– (Nein)	0,0%	0,0	1,0	0,906
GNN5LIN1	5	1	Nein	– (Nein)	0,0%	0,0	1,0	0,906
GNN10LIN1	10	1	Nein	– (Nein)	0,0%	0,0	1,0	0,906

Ende von Tabelle 15

Tabelle 16: Ergebnisliste von $E(kwt)$.

Architektur	GCN-Schichten	Lineare Schichten	Normalisiert	MCC: Median	MCC: CV	TPR	TNR	ACC
	0,0%	1,0	0,932	0,963				
GCN4LIN2	4	2	Ja	0,929	0,5%	1,0	0,931	0,963
GCN7LIN2	7	2	Ja	0,929	0,1%	1,0	0,931	0,963
GCN5LIN4	5	4	Ja	0,929	0,1%	1,0	0,931	0,963
GCN9LIN2	9	2	Ja	0,929	0,1%	1,0	0,931	0,963
GCN4LIN4	4	4	Ja	0,929	12,2%	1,0	0,931	0,963
GCN5LIN2	5	2	Ja	0,929	0,2%	1,0	0,931	0,963
GCN9LIN5	9	5	Ja	0,928	0,1%	1,0	0,931	0,963
GCN6LIN2	6	2	Ja	0,928	0,1%	1,0	0,931	0,963
GCN9LIN3	9	3	Ja	0,928	5,9%	1,0	0,931	0,963
GCN9LIN1	9	1	Ja	0,928	0,2%	0,998	0,932	0,963

Fortsetzung auf der nächsten Seite

Tabelle 16 – Fortsetzung von der letzten Seite

Architektur	GCN-Schichten	Lineare Schichten	Normalisiert	MCC: Median	MCC: CV	TPR	TNR	ACC
GCN6LIN4	6	4	Ja	0,928	0,0%	1,0	0,93	0,963
GCN7LIN4	7	4	Ja	0,928	0,0%	0,999	0,931	0,963
GCN4LIN5	4	5	Ja	0,928	1,3%	1,0	0,93	0,963
GCN6LIN3	6	3	Ja	0,928	0,0%	1,0	0,93	0,963
GCN8LIN4	8	4	Ja	0,928	0,1%	1,0	0,93	0,963
GCN6LIN5	6	5	Ja	0,928	0,1%	1,0	0,93	0,963
GCN2LIN1	2	1	Ja	0,928	0,0%	1,0	0,93	0,963
GCN8LIN2	8	2	Ja	0,928	0,1%	0,999	0,931	0,963
GCN5LIN1	5	1	Ja	0,928	0,1%	0,998	0,932	0,963
GCN6LIN1	6	1	Ja	0,928	0,1%	0,997	0,933	0,963
GCN8LIN3	8	3	Ja	0,928	21,2%	1,0	0,93	0,962
GCN8LIN5	8	5	Ja	0,928	0,0%	1,0	0,93	0,962
GCN5LIN3	5	3	Ja	0,927	0,1%	1,0	0,93	0,962
GCN10LIN5	10	5	Ja	0,927	0,1%	1,0	0,93	0,962
GCN10LIN3	10	3	Ja	0,927	0,1%	1,0	0,93	0,962
GCN3LIN1	3	1	Ja	0,927	0,2%	0,997	0,932	0,962
GCN7LIN5	7	5	Ja	0,927	32,8%	1,0	0,929	0,962
GCN7LIN3	7	3	Ja	0,927	0,1%	0,999	0,931	0,962
GCN9LIN4	9	4	Ja	0,927	0,1%	0,999	0,93	0,962
GCN5LIN5	5	5	Ja	0,926	0,2%	0,999	0,93	0,962
GCN10LIN4	10	4	Ja	0,926	0,1%	0,999	0,929	0,962
GCN4LIN1	4	1	Ja	0,926	2,2%	0,995	0,933	0,962
GCN7LIN1	7	1	Ja	0,924	0,8%	0,993	0,934	0,961
GCN10LIN1	10	1	Ja	0,924	0,3%	0,994	0,932	0,961
GCN8LIN1	8	1	Ja	0,916	0,7%	0,982	0,935	0,957
GCN4LIN3	4	3	Ja	0,913	3,4%	0,995	0,921	0,955
GCN8LIN0	8	0	Ja	0,774	36,9%	1,0	0,762	0,873
GCN6LIN0	6	0	Ja	0,629	16,6%	0,712	0,901	0,813
GCN2LIN0	2	0	Ja	0,487	39,2%	1,0	0,401	0,679
GCN5LIN0	5	0	Ja	0,348	25,0%	0,206	1,0	0,631
GNN10LIN0	10	0	Nein	0,33	0,0%	0,186	1,0	0,622
GNN4LIN5	4	5	Nein	0,33	0,0%	0,186	1,0	0,622
GNN10LIN1	10	1	Nein	0,33	0,0%	0,186	1,0	0,622
GNN6LIN2	6	2	Nein	0,33	0,0%	0,186	1,0	0,622
GNN6LIN1	6	1	Nein	0,33	0,0%	0,186	1,0	0,622
GNN6LIN0	6	0	Nein	0,33	0,0%	0,186	1,0	0,622
GNN5LIN5	5	5	Nein	0,33	0,0%	0,186	1,0	0,622
GNN5LIN4	5	4	Nein	0,33	0,0%	0,186	1,0	0,622
GNN5LIN3	5	3	Nein	0,33	0,0%	0,186	1,0	0,622
GNN5LIN2	5	2	Nein	0,33	0,0%	0,186	1,0	0,622
GNN5LIN1	5	1	Nein	0,33	0,0%	0,186	1,0	0,622
GNN5LIN0	5	0	Nein	0,33	0,0%	0,186	1,0	0,622
GNN4LIN4	4	4	Nein	0,33	0,0%	0,186	1,0	0,622
GNN6LIN5	6	5	Nein	0,33	0,0%	0,186	1,0	0,622
GNN4LIN3	4	3	Nein	0,33	0,0%	0,186	1,0	0,622
GNN4LIN2	4	2	Nein	0,33	0,0%	0,186	1,0	0,622
GNN4LIN1	4	1	Nein	0,33	0,0%	0,186	1,0	0,622
GNN4LIN0	4	0	Nein	0,33	0,0%	0,186	1,0	0,622
GNN3LIN5	3	5	Nein	0,33	51,2%	0,186	1,0	0,622
GNN3LIN4	3	4	Nein	0,33	0,0%	0,186	1,0	0,622

Fortsetzung auf der nächsten Seite

Tabelle 16 – Fortsetzung von der letzten Seite

Architektur	GCN-Schichten	Lineare Schichten	Normalisiert	MCC: Median	MCC: CV	TPR	TNR	ACC
GNN3LIN3	3	3	Nein	0,33	0,0%	0,186	1,0	0,622
GNN3LIN2	3	2	Nein	0,33	0,0%	0,186	1,0	0,622
GNN3LIN1	3	1	Nein	0,33	0,0%	0,186	1,0	0,622
GNN3LIN0	3	0	Nein	0,33	0,0%	0,186	1,0	0,622
GNN2LIN5	2	5	Nein	0,33	70,4%	0,186	1,0	0,622
GNN6LIN4	6	4	Nein	0,33	0,0%	0,186	1,0	0,622
GNN6LIN3	6	3	Nein	0,33	0,0%	0,186	1,0	0,622
GNN7LIN0	7	0	Nein	0,33	0,0%	0,186	1,0	0,622
GNN8LIN5	8	5	Nein	0,33	0,0%	0,186	1,0	0,622
GNN10LIN2	10	2	Nein	0,33	0,0%	0,186	1,0	0,622
GNN10LIN3	10	3	Nein	0,33	0,0%	0,186	1,0	0,622
GNN10LIN4	10	4	Nein	0,33	0,0%	0,186	1,0	0,622
GNN10LIN5	10	5	Nein	0,33	0,0%	0,186	1,0	0,622
GNN2LIN1	2	1	Nein	0,33	0,0%	0,186	1,0	0,622
GNN9LIN5	9	5	Nein	0,33	0,0%	0,186	1,0	0,622
GNN9LIN4	9	4	Nein	0,33	0,0%	0,186	1,0	0,622
GNN9LIN3	9	3	Nein	0,33	0,0%	0,186	1,0	0,622
GNN9LIN2	9	2	Nein	0,33	0,0%	0,186	1,0	0,622
GNN7LIN1	7	1	Nein	0,33	0,0%	0,186	1,0	0,622
GNN9LIN0	9	0	Nein	0,33	0,0%	0,186	1,0	0,622
GNN9LIN1	9	1	Nein	0,33	0,0%	0,186	1,0	0,622
GNN8LIN4	8	4	Nein	0,33	0,0%	0,186	1,0	0,622
GNN8LIN1	8	1	Nein	0,33	0,0%	0,186	1,0	0,622
GNN7LIN2	7	2	Nein	0,33	0,0%	0,186	1,0	0,622
GNN7LIN3	7	3	Nein	0,33	0,0%	0,186	1,0	0,622
GNN7LIN4	7	4	Nein	0,33	0,0%	0,186	1,0	0,622
GNN8LIN0	8	0	Nein	0,33	0,0%	0,186	1,0	0,622
GNN7LIN5	7	5	Nein	0,33	0,0%	0,186	1,0	0,622
GCN4LIN0	4	0	Ja	0,33	1,4%	0,186	1,0	0,622
GNN8LIN2	8	2	Nein	0,33	0,0%	0,186	1,0	0,622
GNN8LIN3	8	3	Nein	0,33	0,0%	0,186	1,0	0,622
GCN9LIN0	9	0	Ja	0,33	47,2%	0,186	1,0	0,622
GCN7LIN0	7	0	Ja	0,33	0,0%	0,186	1,0	0,622
GCN10LIN0	10	0	Ja	0,33	48,3%	0,186	1,0	0,622
GNN2LIN0	2	0	Nein	0,326	0,0%	0,186	0,998	0,621
GCN3LIN0	3	0	Ja	0,325	0,0%	0,186	0,998	0,621
GNN2LIN4	2	4	Nein	0,029	83,9%	1,0	0,002	0,466
GNN2LIN3	2	3	Nein	0,029	98,8%	1,0	0,002	0,466
GNN2LIN2	2	2	Nein	0,029	0,0%	1,0	0,002	0,466

Ende von Tabelle 16