LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
LEHRSTUHL FÜR MOBILE UND VERTEILTE SYSTEME

Trends in Mobilen und Verteilten Systemen

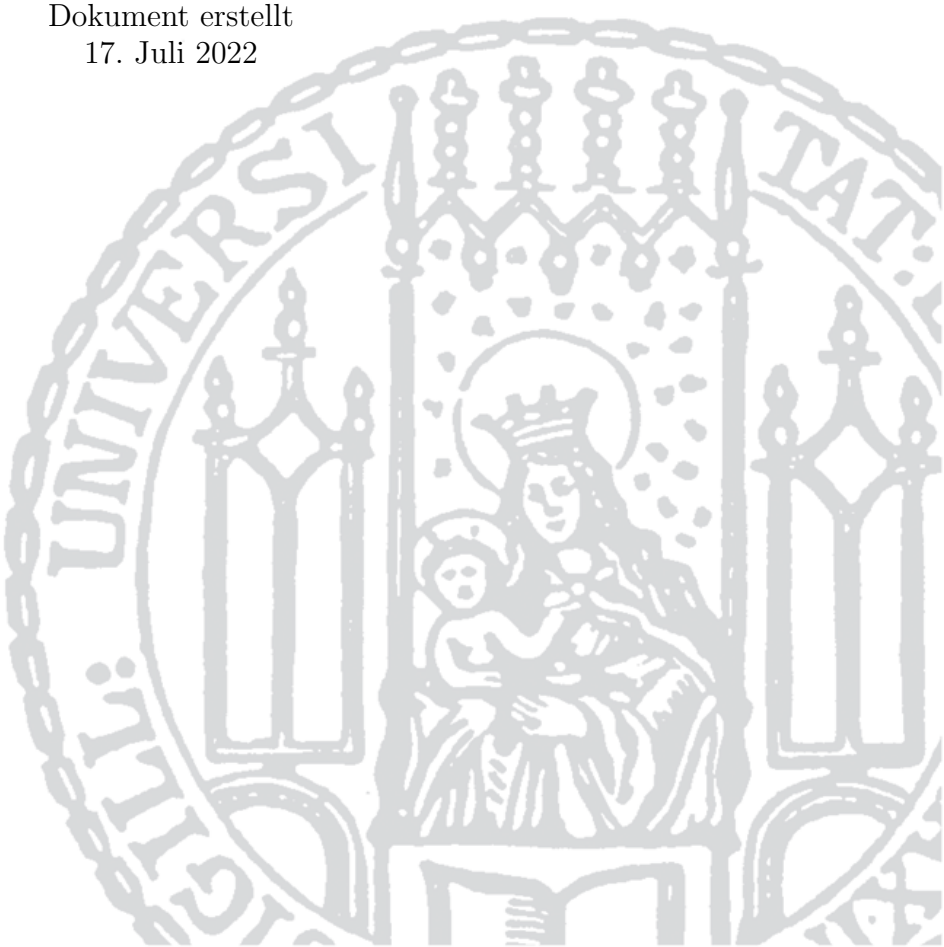
SATNet: Integration eines Erfüllbarkeitslösers in Deep Learning Architekturen

Bearbeiter: Marcus Reiners

Betreuer: Jonas Nüßlein

Aufgabensteller: Prof. Dr. Claudia Linnhoff-Popien

Dokument erstellt
17. Juli 2022



SATNet: Integration eines Erfüllbarkeitslösers in Deep Learning Architekturen

Zusammenfassung

In dieser Arbeit wird die Funktionsweise eines Erfüllbarkeitslösers als Schicht eines neuronalen Netzwerks präsentiert und anhand von Experimenten das Erlernen logischer Zusammenhänge, im Zusammenschluss mit größeren Deep Learning Architekturen, veranschaulicht.

Inhaltsverzeichnis

1 Einleitung

Deep Learning hat bereits in der Vergangenheit bedeutende Fortschritte gemacht, die allesamt wichtige Schritte beim Ausarbeiten einer künstlichen Intelligenz darstellen. Jedoch werden die Lernprozesse innerhalb eines neuronalen Netzwerks oft nur aufbauend von bereits eingespeistem Vorwissen durchgeführt. Somit muss ein neuronales Netz immer gewisse Vorbedingungen kennen, um Problemstellungen zu lösen und kann dabei nicht, wie das menschliche Gehirn, logische Schlüsse ziehen. Betrachtet wird dieses Problem anhand von Erfüllbarkeitsproblemen, welche innerhalb eines tiefer gehenden neuronalen Netzwerks mithilfe eines MAXSAT-Lösers, das Lernen von logischen Zusammenhängen erlauben. Mithilfe dieses Ansatzes, lassen sich somit Problemstellungen lösen, die vorher unmöglich, ohne jegliches Vorwissen an logischen Zusammenhängen von Deep Learning Architekturen, lösbar waren. Die Effektivität dieses Systems wird anhand von drei Beispielen gezeigt, dazu gehört das Lösen von Paritätsfunktionen mit einer Einzelbit-Vorschau und das Lösen eines 9x9 Sudokus, welches zusätzlich noch in handschriftlicher Form vorliegt und dementsprechend mithilfe eines integriertem Zahlen-Erkennungsnetzwerks von einer größeren Architektur gelöst wird. Zusammenfassend zeigen diese Beispiele einen großen Schritt in Richtung logischem Schließen innerhalb Deep Learning Architekturen.

2 Ein differenzierbarer Erfüllbarkeitslöser

2.1 SATNet

Das MAXSAT-Problem stellt die optimierte Variante des SAT-Problems dar, bei welchem überprüft wird, ob ein logischer Ausdruck lösbar ist. Zum Lösen eines MAXSAT-Problems, wird also die maximale Anzahl an logisch lösbaren Ausdrücken verlangt. Dieses Problem lässt sich darstellen als:

$$\sum_{j=1}^{m} \bigvee_{i=1}^{n} \mathbf{1}\left\{\tilde{s}_{ij}\tilde{v}_i > 0\right\} \qquad (1),$$

wobei n der Variablenanzahl und m der Klauselnanzahl einer MAXSAT-Instanz entspricht. Sei $\tilde{v}_i \in \{-1, 1\}^n$ der Wahrheitswert der Variable $i \in \{1, ..., n\}$, und sei $\tilde{s}_i \in \{-1, 0, 1\}^m$ mit $i \in \{1, ..., m\}$, wobei $\tilde{s}_{ij}$ das Vorzeichen von $\tilde{v}_i$ in der Klausel $j \in \{1, ..., m\}$ bestimmt. Es wird in jeder Klausel überprüft, ob der Wahrheitswert *wahr* vorliegt, folglich wird dann eine **1** aufsummiert. Eine SATNet Schicht nimmt nun als Eingabe diskrete oder probabilistische Zuordnungen von bekannten MAXSAT Variablen und gibt Vermutungen, mithilfe einer MAXSAT SDP Relaxation[1] wieder, was in (Abbildung 1) veranschaulicht wird. [3, p. 2-3]

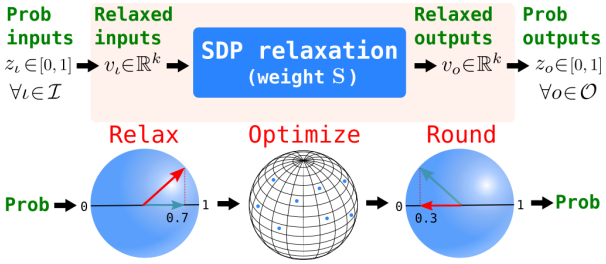


Abbildung 1: SATNet forward pass [2]

2.2 Forward pass

Sei $Z_\mathcal{I}$ die Menge aller probabilistischen oder binären Variablen bekannter Zuweisungen mit $z_i \in [0,1]$ und $i \in \mathcal{I}$, mit $\mathcal{I} \subset \{1,...,n\}$. $Z_\mathcal{O}$ sei die Menge aller probabilistischen oder (optional, zur Testzeit) binären Variablen mit unbekannten Zuweisungen, mit $z_o \in [0,1]$, und $o \in \mathcal{O}$, mit $\mathcal{O} \equiv \{1,...,n\} \setminus \mathcal{I}$. Dabei stellen $Z_\mathcal{O}$ die aus den Inputs $Z_\mathcal{I}$ über SDP Relaxation erzeugten Outputs dar. [3, p. 3]

2.2.1 Schritt 1: Relaxation der Inputs

Damit die Inputs $Z_\mathcal{I}$ als SDP Relaxation anwendbar sind, werden die Inputs z_i mit $i \in \mathcal{I}$ zu zufällig generierten Einheitsvektoren $v_i \in \mathbb{R}^k$ relaxiert, wobei mit $k = \sqrt{2n}+1$ und mit n als Anzahl aller Problemvariablen, die optimale Lösung der MAXSAT Relaxation erzielt wird. Die Gleichung

$$v_i^T v_\top = -cos(\pi z_i) \qquad (2),$$

wird mit

$$v_i = -cos(\pi z_i) v_\top + sin(\pi z_i)(I_k - v_\top v_\top^T) v_i^{\text{rand}} \qquad (3)$$

erfüllt. [3, p. 4]

2.2.2 Schritt 2: Erzeugung relaxierter Outputs

Mithilfe des Koordinatenabstiegs werden Werte für die unbekannten Output Variablen errechnet, dabei besitzt jeder Durchlauf eine Laufzeit von $O(\text{nmk})$. [3, p. 4]

2.2.3 Schritt 3: Erzeugung diskreter/probabilistischer Outputs

Die aus dem Koordinatenabstieg relaxierten Outputs werden mithilfe von zufälligem Runden in die diskreten oder probabilistischen Variablenzuweisungen $Z_\mathcal{O}$ umgewandelt.

$$\tilde{v}_o = \begin{cases} 1 & \text{wenn } sign(v_o^T\ r) = sign(v_\top^T\ r) \\ -1 & \text{andererseits} \end{cases}, o \in \mathcal{O} \qquad (4)$$

r stellt hierbei eine zufällige Ebene aus der Einheitssphäre dar. Während der Trainingsphase wird kein zufälliges Runden verwendet, hierbei gilt $z_o = P(\tilde{v}_o)$, mit

$$P(\tilde{v}_o) = cos^{-1}(-v_o^T v_\top)/\pi \qquad (5).$$

Während der Testphase kann z_o auf die gleiche Art, oder mithilfe von zufälligem Runden, wie in (4), ermittelt werden. [3, p. 4]

2.3 Backpropagation

Mit den Gradienten $\frac{\partial l}{\partial Z_\mathcal{O}}$, mit l als loss des Netzwerks und in Abhängigkeit zu den Outputs $Z_\mathcal{O}$, lassen sich die Gradienten $\frac{\partial l}{\partial Z_\mathcal{I}}$ in Abhängigkeit der Inputs und $\frac{\partial l}{\partial Z_\mathcal{S}}$ in Abhängigkeit der Gewichte errechnen, um die Anpassungen der Layer Inputs an den entscheidenden Stellen zu tätigen. [3, p. 4-5]

2.3.1 Schritt 1: Von probabilistischen zu realxierten Outputs

Den Gradienten $\frac{\partial l}{\partial V_\mathcal{O}}$, in Abhängigkeit zu den realxierten Outputs erhält man, in dem man ihn als $(\frac{\partial l}{\partial Z_\mathcal{O}}) \cdot (\frac{\partial l}{\partial V_\mathcal{O}})$ umschreibt, wobei sich $\frac{\partial l}{\partial V_\mathcal{O}}$ durch die Ableitung der umgeformten Gleichung aus (5), nämlich als $cos(\pi z_o) = -v_\top^T v_o$, errechnen lässt. So erhält man

$$\frac{\partial l}{\partial V_\mathcal{O}} = \left(\frac{\partial l}{\partial Z_\mathcal{O}}\right)\left(\frac{\partial z_o}{\partial V_\mathcal{O}}\right) = \left(\frac{\partial l}{\partial Z_\mathcal{O}}\right)\frac{1}{\pi sin(\pi z_o)} \qquad (6).$$

[3, p. 5]

2.3.2 Schritt 2: Von relaxierten Outputs zu relaxierten Inputs

Um von den Gradienten $\frac{\partial l}{\partial V_\mathcal{O}}$ zu den Gradienten $\frac{\partial l}{\partial V_\mathcal{I}}$ zu gelangen, wird wie beim forward pass die Prozedur von SDP angewandt. [3, p. 5]

2.3.3 Schritt 3: Von relaxierten zu ursprünglichen Inputs

Um zu den Gradienten $\frac{\partial l}{\partial Z_\mathcal{I}}$ in Abhängigkeit der ursprünglichen Inputs zu gelangen, werden die Gradienten $\frac{\partial l}{\partial V_\mathcal{I}}$ durch den Schritt 1 aus 3.2.1 geführt, indem Gleichung (3) abgeleitet wird. Man erhält

$$\frac{\partial l}{\partial z_i} = \frac{\partial l}{\partial z_i^*} + \left(\frac{\partial l}{\partial v_i}\right)^T \frac{\partial v_i}{\partial z_i} \qquad (7),$$

wobei

$$\frac{\partial v_i}{\partial z_i} = \pi(sin(\pi z_i)v_\top + cos(\pi z_i)(I_k - v_\top v_\top^T)v_i^{\text{rand}} \qquad (8).$$

[3, p. 5]

3 Experimente

3.1 Lernen von Parität (verkettete XOR)

Dieses Experiment testet das Erlernen SATNets von Parität mit Einzelbit-Vorschau. Dabei soll aus Eingabesequenzen, bestehend aus Bitfolgen die zugehörige Parität ermittelt werden. Das Lernen von Parität war für bisherige konventionelle Deep Learning Architekturen schwer lösbar, aber da das Paritäts-Problem in NP liegt, lässt es sich auf SAT reduzieren. Das Modell ist dabei so aufgebaut, dass es bei einer Länge von L eine Anzahl von L - 1 SATNet Schichten, mit gebundenen Gewichten enthält. Dabei nimmt die erste Schicht die ersten beiden Binärwerte der Bitfolge als Input. Die Schicht d nimmt dabei den Binärwert d und zusätzlich den gerundeten Output der Schicht d - 1 als Input. Sobald jede einzelne Schicht das Berechnen von XOR-Funktionen erlernt hat, wir das gesamte Modell die korrekte Parität ausgeben. Um diese Ausgaben zu erlangen muss das System erst eine lange Abfolge von SAT-Problemen bearbeiten. Für die vorgegebenen Längen von $L = 20$ und $L = 40$ wird ein zufällig generierter Datensatz von zehntausend Beispielen erzeugt, 9000 davon werden dem System als Trainingseinheit gegeben und 1000 schließlich als Testeinheit. Verglichen mit einem LSTM-Sequenzklassifizierer erzielt das SATNet hierbei deutlich bessere Ergebnisse (Bild 2). LSTM ist nicht dazu in der Lage eine angemessen Darstellung der Parität zu lernen und hat auch im Laufe von 100 Trainingsepochen keine signifikante Verbesserung erzielt. Über beide Input-Längen erzielt LSTM dabei nur eine Testerrorrate von bestenfalls 0,476 (Ein zufälliges Raten hätte eine Rate von 0,5). [3, p. 6]

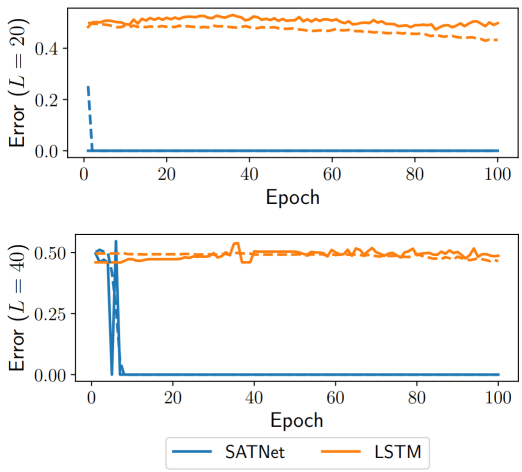


Abbildung 2: Fehlerrate des Paritätsexperiments [3, p. 6]

3.2 Sudoku (original und permutiert)

Sobald die Regeln von Sudoku bekannt sind, lässt sich das Problem rechnerisch leicht bewältigen. Die eigentliche Herausforderung stellt dabei das Erlernen der Regeln dar, was ebenfalls für konventionelle neuronale Netzwerke Probleme dargestellt hat. Normalerweise wurden Sudoku Problemstellungen über Baumsuchen gelöst, diese lassen sich jedoch schlecht in neuronalen Netzen Ausführen, sind aber leicht als SAT oder MAXSAT Probleme darzsutellen. Dem SATNet wird hierbei als Input eine logische Bitdarstellung geliefert, welche zudem die zu lernenden Bits, also die unbekannten Felder im Spielfeld, darstellt. Der Output besteht folglich wieder aus einer Bitdarstellung mit zufällig Gewählten Eingaben für die unbekannten Bits. Erneut wurden für dieses Experiment 9000 Trainingsbeispiele und 1000 Testbeispiele erzeugt, welche bei SATNet nach 100 Trainingsepochen zu einer Genauigkeit von 98,3% führten. Verglichen mit dem faltenden neuronalen Netzwerk "ConvNet", löst dieses die Trainingseinheiten mit einer Genauigkeit von 72,6%, die eigentlichen Testeinheiten jedoch nur mit 0,04%. Um sich nun zu vergewissern, dass SATNet nicht einfach nur gewisse Strukturen in der Eingabe erkennt, sondern tatsächlich die Sudokuregeln lernt, wurde das Experiment mit einer permutierten Version, bei der die Eingaben entsprechend angepasst wurden, erneut ausgeführt. Durch die Permutation liegen keine erkennbaren Strukturen innerhalb der Eingabe vor, die Spielregeln von Sudoku und die logischen Zusammenhänge der Eingabe bleiben jedoch bestehen. SATNet hat unter diesen Bedingungen erneut eine Genauigkeit von 98,3% erzielt und zeigt somit, dass diese Architektur dazu in der Lage ist die Spielregeln zu erlernen und anzuwenden. ConvNet hingegen erzielte bei dem permutierten Test eine null prozentige Genauigkeit. [3, p. 6-7]

3.3 Visuelles Sudoku

Um nun die Integration von SATNet in größeren Architekturen zu zeigen, werden in einem Experiment die SATNet-Schichten mit denen eines faltenden neuronalen Netzwerks kombiniert, welches für die Erkennung eines visuell vorliegenden Sudoku-Spielfeld nötig ist. Der Output dieser Schichten wird SATNet als logischer Input geliefert, wie in 4.2. Hierzu findet wieder ein Vergleich zu einer ConvNet Architektur statt, wobei der Ablauf der Zahlenerkennung gleich verläuft. Unter denselben Trainingsbdingungen kann eine maximale Genauigkeit von 74,7% erreicht werden, was der einhergehenden visuellen Erkennung verschuldet ist. Nach 100 Epochen erzielt die SATNet Architektur eine Genauigkeit von 63,2%, was 85% des theoretischen Optimums entsprechen. Das ConvNet besitzt nach 100 Epochen keine nennenswerte Genauigkeit und kann keine sinnvollen Lösungen liefern. So ist es der gesamten Architetkur, dank dem SATNet auch möglich die Sudokuregeln von einer handschriftlich vorliegenden Repräsentation zu lernen, was zuvor nicht möglich war. [3, p. 7-8]

4 Folgerung

In dieser Arbeit wird die Funktionsweise der SATNet Schicht präsentiert, die dazu in der Lage ist Erfüllbarkeitsprobleme zu lösen, wie in den Experimenten der Parität und dem Erlernen der Sudoku Spielregeln veranschaulicht wird. Zudem wurde mit dem Experiment des visuellen Sudokus gezeigt, dass sich diese Schicht auch in größere Deep Learning Architekturen integrieren lässt, was diesen die Möglichkeit gibt, logische Zusammenhänge zu erlernen, ohne dabei auf hinzugegebene Informationen zurückzugreifen. Das tatsächliche Erlernen von logischen Zusammenhängen konnte in den bisherigen Deep Learning Architekturen nicht überzeugend umgesetzt werden. SATNet zeigt jedoch hierbei große Fortschritte im Erlernen dieser Zusammenhänge und stellt dabei einen wichtigen Teil des Deep Reasoning dar.

Autorenschaft

Marcus Reiners hat diese Seminararbeit mit dem Titel "SATNet: Integration eines Erfüllbarkeitslösers in Deep Learning Architekturen-verfasst. In einer wissenschaftlichen Arbeit ohne Prüfungszweck stünde hier eine Danksagung!

Literatur

[1] Po-Wei Wang, Wei-Cheng Chang, and J. Zico Kolter. The mixing method: low-rank coordinate descent for semidefinite programming with diagonal constraints, 2017.

[2] Po-Wei Wang, Priya L. Donti, Bryan Wilder, and J. Zico Kolter. Satnet. `https://github.com/locuslab/SATNet`, 2019.

[3] Po-Wei Wang, Priya L. Donti, Bryan Wilder, and J. Zico Kolter. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver, 2019.