Lernen mit differenzierbaren Algorithmen[1]

Felix Petersen[2]

Abstract: Klassische Algorithmen und maschinelle Lernsysteme wie neuronale Netze begegnen uns beide häufig im Alltag. Während klassische Algorithmen für die präzise Ausführung genau definierter Aufgaben wie dem Finden des kürzesten Wegs in einem Graphen geeignet sind, ermöglichen neuronale Netze das Lernen aus Daten, um die wahrscheinlichste Antwort in komplexeren Aufgaben wie der Bildklassifizierung vorherzusagen. Um das Beste aus beiden Welten zu vereinen, kombiniert diese Arbeit beide Konzepte, was zu robusteren, leistungsfähigeren, interpretierbareren, recheneffizienteren und dateneffizienteren Architekturen führt. Bei der Integration eines Algorithmus in eine neuronale Architektur ist es wichtig, dass der Algorithmus differenzierbar ist, sodass die Architektur Ende-zu-Ende trainiert werden kann. Um Algorithmen differenzierbar zu machen, präsentiert diese Arbeit ein allgemeines Verfahren zur stetigen Relaxierung von Algorithmen. Überdies präsentiert diese Arbeit konkrete differenzierbare Algorithmen wie differenzierbare Sortier-Netzwerke, differenzierbare Renderer, und differenzierbare Logik-Gatter-Netzwerke.

1 Einleitung

Vor 4000 Jahren erfanden die Ägypter einen Algorithmus zur Multiplikation von zwei Zahlen, die früheste Aufzeichnung eines Algorithmus [Ne69]. 1843 veröffentlichte Ada Lovelace den ersten Computer-Algorithmus und sah moderne Anwendungen von Computern wie Kunst und Musik voraus, zu einer Zeit, als ein solcher Computer noch nicht einmal gebaut war [ML43]. Ein Jahrhundert später, 1943, entwickelten McCulloch und Pitts [MP43] das erste mathematische Modell von neuronalen Netzen aufgrund von Beobachtungen biologischer Prozesse im Gehirn. In den letzten Jahren haben sich Ansätze basierend auf künstlichen neuronalen Netzwerken in der Forschung großer Beliebtheit erfreut. Dieser Wiederaufschwung kann auf Fortschritte in der Hardware [KWM16], Software [Pa19], der Entwicklung von CNNs [Fu80, Le99] und der Effektivität von Deep Learning bei vielen Aufgaben (z.B. Bildklassifizierung [KSH12]) zurückgeführt werden.

Sowohl klassische Algorithmen als auch maschinelle Lernsysteme wie neuronale Netze sind mittlerweile im Alltag allgegenwärtig. Klassische Algorithmen sind besonders für die präzise Ausführung genau definierter Aufgaben wie dem Finden des kürzesten Wegs in einem großen Graphen geeignet. Neuronale Netze ermöglichen das Lernen aus Daten, um die wahrscheinlichste Antwort in komplexeren Aufgaben wie der Bildklassifizierung vorherzusagen — Aufgaben, die nicht auf einen exakten Algorithmus reduziert werden können. Um das Beste aus beiden Welten zu vereinen, wird in der hier zusammengefassten Dissertation [Pe22a] die Kombination von *klassischen Informatik-Algorithmen* und *neuronalen Netzen*, oder allgemeiner, maschineller Lernsysteme, untersucht. Dies führt zu

[1] Englischer Titel der Dissertation: “Learning with Differentiable Algorithms”

[2] Stanford University, USA. mail@felix-petersen.de

robusteren, besser performenden, interpretierbareren, recheneffizienteren und dateneffizienteren Architekturen. Hierbei kann Robustheit eines Modells erreicht werden, indem ein beweisbar korrekter Algorithmus auf Vorhersagen eines neuronalen Netzes angewendet wird. Gleichzeitig kann die Effizienz eines Modells erhöht bzw. die Rechenkomplexität reduziert werden, indem ein Teil eines neuronalen Netzwerkes durch einen schnelleren Algorithmus ersetzt wird. Auch hinsichtlich der Genauigkeit kann das Modell verbessert werden, da durch den Einsatz eines Algorithmus das Risiko von Fehlern reduziert werden kann und das Domain-Wissen des Algorithmus das Modell unterstützt. Entsprechend können diese Modelle auch interpretierbarer sein, da die Eingabe zu Algorithmen (im Gegensatz zu versteckten Schichten) interpretierbar ist. Die algorithmische Überwachung, welche in der Dissertation formalisiert wird, ist eine Form von schwach überwachtem Lernen, wodurch die Modelle daten-/label-effizienter sind.

Normalerweise werden neuronale Netze mit stochastischem Gradientenabstieg (SGD) oder präkonditionierten SGD-Methoden wie dem Adam-Optimierungsverfahren [KB15] trainiert. Diese Methoden basieren auf der Berechnung des Gradienten (d.h. der Ableitung) einer Verlustfunktion in Bezug auf die Parameter des Modells. Dieser Gradient zeigt die Richtung des steilsten Anstiegs der Verlustfunktion. Da das Minimieren der Verlustfunktion das Modell verbessert, kann das Modell optimiert werden, indem die Parameter in die entgegengesetzte Richtung des Gradienten bewegt werden, also ein Gradientenabstieg. Die Ableitung der Verlustfunktion bezüglich der Modellparameter kann effizient mithilfe des Rückpropagationsverfahrens [RHW86] berechnet werden, der in den heutigen Deep-Learning-Frameworks [Pa19] als rückwärtsgerichtete automatische Differenzierung implementiert ist.

Gradientenbasiertes Lernen erfordert, dass alle beteiligten Operationen differenzierbar sind; jedoch sind viele interessante Operationen wie Sortieralgorithmen nicht differenzierbar. Das liegt daran, dass bedingte Verzweigungen wie `if` stückweise konstant sind, d.h. sie haben eine Ableitung von 0, mit Ausnahme der Übergänge (d.h. “Sprünge”) zwischen `true` und `false`, bei denen ihre Ableitung nicht definiert ist. Dementsprechend ist gradientenbasiertes Lernen mit (nicht-differenzierbaren) Algorithmen im Allgemeinen nicht möglich. Daher konzentriert sich die Dissertation darauf, Algorithmen durch kontinuierliche Relaxierung und Approximationen differenzierbar zu machen, um sie in neuronale Netze zu integrieren und so die Vorteile beider Ansätze zu nutzen. Die Grundidee der kontinuierlichen Relaxierung ist, ein Maß an Unsicherheit in einen Algorithmus einzuführen, was z.B. zu glatten Übergängen zwischen `true` und `false` bei `if`-Anweisungen führen kann, wodurch der Algorithmus vollständig differenzierbar wird.

Das Lernen mit differenzierbaren Algorithmen kann in zwei Disziplinen unterteilt werden:

- Differenzierbare Algorithmen, d.h. die Untersuchung, wie durch Algorithmen zurückpropagiert werden kann und sinnvolle Gradienten bestimmt werden können.
- Algorithmische Überwachung, d.h. die Integration von Algorithmen und algorithmischem Wissen in das Lernen von neuronalen Netzwerken.

Die folgenden beiden Abschnitte behandeln *differenzierbare Algorithmen* und die *algorithmische Überwachung*.

2 Differenzierbare Algorithmen

Im Allgemeinen besteht die Herausforderung beim Ende-zu-Ende Training von neuronalen Architekturen mit integrierten Algorithmen darin, die Gradienten des jeweiligen integrierten Algorithmus zu bestimmen, z.B. durch eine differenzierbare Approximation.

Die Kernidee hinter den meisten differenzierbaren Algorithmen ist die Glättung, d.h. die Störung durch eine Wahrscheinlichkeitsverteilung. Differenzierbare Algorithmen können in zwei große methodologische Schulen unterteilt werden:

- Glättung durch stochastische Störungen durch Sampling.
- Glättung durch analytische Störungen, die in geschlossener Form gelöst werden.

Der primäre Fokus der Dissertation liegt auf analytischen Störungen, die in geschlossener Form gelöst werden, behandelt jedoch auch stochastische Sampling-Methoden.

Diese Störungen können an zwei verschiedenen Stellen eingeführt werden, was zu zwei konzeptionell verschiedenen mathematischen Modellannahmen führt:

- Störungen der Eingaben eines Algorithmus.
- Störungen von Variablen / Bedingungen in einem Algorithmus.

Das Glätten der Eingaben eines Algorithmus bedeutet, einen Algorithmus $f(x)$ zu glätten zu $\mathbb{E}[f(x+\varepsilon)]$, wobei ε aus einer bestimmten Verteilung gezogen wird (z.B. der Gauß-Verteilung). Dementsprechend erfordert das Glätten der Eingaben eines Algorithmus in der Regel keine Annahmen über einen Algorithmus und kann mit stochastischer Glättung gelöst werden: Diese Methode ist unabhängig vom verwendeten Algorithmus, vorausgesetzt, der Algorithmus ist korrekt und löst das entsprechende Problem.

Auf der anderen Seite entspricht das Glätten von Variablen und Bedingungen in einem Algorithmus dem Relaxieren von Aussagen wie `if` x `> 0` zu `if` $x+\varepsilon$ `> 0`. Somit kann ein Ausdruck wie z.B. `a = y if` x `> 0 else z` relaxiert und wie folgt in geschlossener Form berechnet werden (z.B. für ε aus einer logistischen Verteilung mit CDF σ): `a` $= \sigma(x) \cdot$ `y` $+ (1-\sigma(x)) \cdot$ `z`. In diesem Fall hängt die resultierende Funktion von der konkreten Wahl des Algorithmus für ein bestimmtes Problem ab. Das bedeutet, dass bei einer guten Wahl des Algorithmus (z.B. Bellman-Ford) das Glätten von Variablen und Bedingungen zu besseren Entspannungen führen kann als das Glätten von Eingaben; jedoch bei einer schlechten oder unangemessenen Wahl des Algorithmus (z.B. Dijkstra) ein unreflektierendes Glätten von Eingaben besser funktionieren würde.

Während sich das Glätten von Eingaben für stochastische Glättung eignet und für analytische Propagationsmethoden unlösbar sein kann, ist das Glätten von Variablen mit stochastischen Methoden schwierig, während es mithilfe analytischer Propagationsmethoden ausgewertet werden kann. Die Beiträge in der hier zusammengefassten Arbeit konzentrieren sich in erster Linie auf das Glätten von Variablen in einem Algorithmus; das Glätten von Eingaben wird jedoch auch untersucht.

	Variablen	Eingaben
analytisch	[Pe21a] [Pe21b] [Pe22d] [Pe22c] [Pe22e] [Pe19] [Li19] [Ch19]	([CSL10])
stochastisch	[Li21]	[ALT16] [Be20] [Ni19] [Co21]

Abb. 1: Klassifikation einer Auswahl differenzierbarer Algorithmen in analytische vs. stochastische Berechnung und Modellannahmen von Störungen der Variablen vs. Eigaben. ([CSL10] bestimmt keine Gradienten.)

Abb. 1 klassifiziert eine Auswahl von auf Störungen basierenden differenzierbaren Algorithmen hinsichtlich der Methode (analytische vs. stochastische Störungen) und Annahmen (Störungen von Eingaben vs. Störungen von Variablen). Die meisten Methoden nutzen entweder analytische Störungen von Variablen oder stochastische Störungen von Eingaben. Im Folgenden werden diese Klassifikation und Alternativen anhand einiger Beispiele diskutiert:

Im Bereich des differenzierbaren Renderings [Pe19, Ka20, Pe22f] ist die Methode der annähernden analytischen Störungen beliebt, da Methoden wie die stochastische Glättung aufgrund der hohen Dimension des betroffenen Raums (Bildraum und Raum von 3D-Modellen) typischerweise unlösbar sind. Entsprechend modellieren diese Methoden Störungen von Variablen. Das Modellieren von Eingabestörungen mit analytischen Methoden wäre aufgrund der Komplexität der Rendering-Algorithmen unlösbar. Die hier zusammengefasste Arbeit liefert Indizien, dass Störungen der Variablen besser sind als Störungen der Eingaben, wenn ein adäquater Algorithmus für ein bestimmtes Problem verfügbar ist. Es gibt auch differenzierbare Renderer, die auf Monte-Carlo-Sampling basieren [Li18, Ni19, Zh21] und Lidec *et al.* [Li21] modellieren Störungen der Variablen mit der auf Sampling basierenden stochastischen Glättungsmethode. Im Bereich der differenzierbaren Sortierung gibt es Methoden, die alternative Methoden zur Glättung verwenden. Beispiele hierfür sind die heuristischen differenzierbare Sortieralgorithmen wie NeuralSort [Gr19] und SoftSort [PE20]. Eine weitere Methode, von Cuturi *et al.* [CTV19] vorgestellt, reduziert die Sortierung auf optimalen Transport (OT) und führt eine entropische Regularisierung ein, die das OT-Problem relaxiert und dadurch die Sortierung differenzierbar macht.

3 Algorithmische Überwachung

Künstliche neuronale Netze haben ihre Fähigkeit gezeigt, verschiedene Probleme zu lösen, von klassischen Aufgaben in der Informatik, wie maschinellem Übersetzen [Va17] und Objekterkennung [Re16], bis hin zu vielen anderen Themen in der Wissenschaft, wie Protein-Faltung [Se20]. Gleichzeitig gibt es klassische Algorithmen, welche typischerweise bestimmte Aufgaben anhand einer vordefinierten Steuerstruktur lösen, wie Sortieren oder das Berechnen des kürzesten Weges und für welche Garantien über ihr Verhalten abgeleitet werden können. Kürzlich hat die Forschung begonnen, beide Elemente durch die Integration algorithmischer Konzepte in neuronale Netzarchitekturen zu kombinieren.

Diese Ansätze ermöglichen es, neuronale Netze mit alternativen Überwachungs-Strategien zu trainieren, wie z.B. das Lernen von 3D-Darstellungen über einen differenzierbaren Ren-

derer oder das Trainieren von neuronalen Netzen mit Ordnungsinformationen. Die zusammengefasste Arbeit vereinigt diese alternativen Überwachungs-Strategien, welche Algorithmen in das Trainingsziel integrieren, als algorithmische Überwachung:

Definition 1 (Algorithmische Überwachung)**.** *Bei der algorithmischen Überwachung wird ein Algorithmus auf die Vorhersagen eines Modells angewandt und die Ausgaben des Algorithmus werden überwacht. Im Gegensatz dazu werden bei der direkten Überwachung die Vorhersagen eines Modells direkt überwacht. Dies wird in Abb. 2 veranschaulicht.*

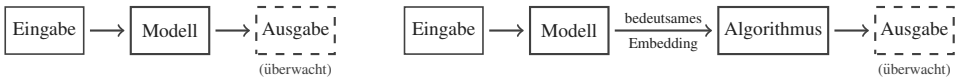

Abb. 2: Direkte Überwachung (links) im Vergleich zu algorithmischer Überwachung (rechts).

4 Beiträge der Dissertation

Allgemeine differenzierbare Algorithmen. Die Dissertation präsentiert eine allgemeine Methode für die analytische Glättung von Algorithmen über Störung der Variablen. Hierfür werden alle Variablen im Algorithmus, nach denen abgeleitet werden soll, durch logistische Verteilungen gestört. Die Methode wird auf die folgenden algorithmischen Überwachungs-Probleme angewandt: Sortier- und Ordnungsüberwachung, kürzester-Pfad-Überwachung, Silhouetten-Überwachung und Editierdistanzüberwachung. Auf diesen Problemen erzielt die allgemeine Methode, welche auch auf der NeurIPS 2021 veröffentlicht wurde, Verbesserungen im Vergleich zu existierenden spezialisierten Methoden.

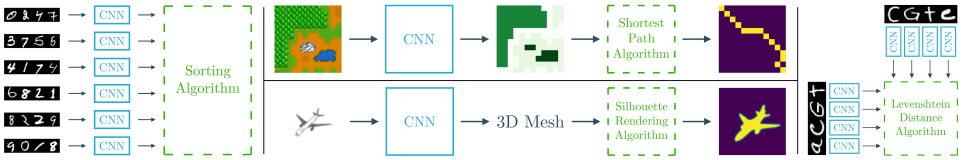

Abb. 3: Übersicht über die algorithmischen Überwachungsexperimente: Ordnungsüberwachung (links), kürzester-Pfad-Überwachung (oben mitte), Silhouetten-Überwachung (unten mitte) und Editierdistanzüberwachung (rechts).

Differenzierbares Sortieren und Ranking. Sortier- und Rankingalgorithmen haben eine lange Tradition als wahrscheinlich einer der am häufigsten untersuchten Klassen von Algorithmen in der Informatik. Dementsprechend ist es naheliegend, Sortieralgorithmen in neuronale Netzwerkarchitekturen integrieren zu wollen. Während die Arbeit bereits eine generelle Methode vorstellt, welche z.B. den Bubble-sort Sortieralgorithmus differenzieren kann, wird der Fall des Sortierens in weiterem Detail studiert. Konkret werden differenzierbare Sortiernetzwerke [Pe21a] vorgestellt (ICML 2021), welche sich an dem Informatik-Konzept der Sortiernetzwerke [Kn98] orientieren. Diese erreichten Ergebnisse auf dem neuesten Stand der Technik für den Sorting MNIST Benchmark. Zudem wurde der Sorting SVHN Benchmark in dieser Arbeit eingeführt. Darauf aufbauend, wurden differenzierbare Sortiernetzwerke mit speziellen theoretischen Garantien entwickelt [Pe22d] (ICLR 2022). Konkret erzeugt die Erweiterung monotone, Lipschitz-stetige und fehler-

beschränkte differenzierbare Sortiernetzwerke durch eine spezielle Wahl an Wahrscheinlichkeitsverteilungen, durch welche die Variablen gestört werden. Die Monotonität ist von besonderem Vorteil, da Monotonität in Optimierungsprozessen lokale Minima verhindert und somit die Genauigkeit verbessert. Die Beschränkung des Fehlers erlaubt es, eine differenzierbare Sortierfunktion zu erhalten, welche zwar Störungen durch eine Verteilung benötigt, damit sie glatt und differenzierbar ist, jedoch gleichzeitig nur eine minimale Abweichung von der originalen harten Sortierfuntion aufweist, was auch die Genauigkeit verbessert. Die Arbeit demonstriert, dass diese Fortschritte die Genauigkeit auf den üblichen Benchmarks noch einmal signifikant verbessern. Auf den differenzierbaren Sortiernetzen aufbauend wurden u.a. auch Methoden für selbst-überwachtes Lernen entwickelt [Sh23].

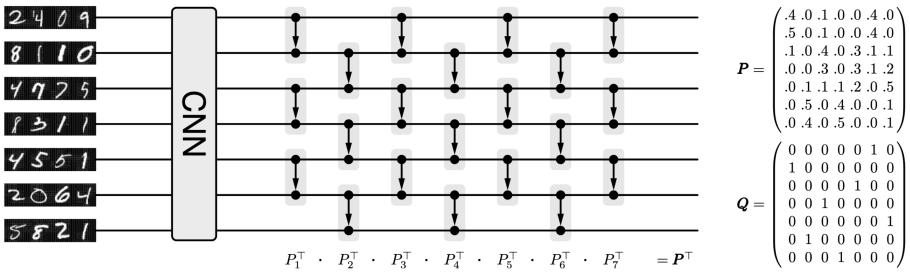

Abb. 4: Überblick über das System für das Training mit Sortierüberwachung. Links: Die Eingabebilder werden getrennt und unabhängig voneinander in ein CNN eingespeist, welches sie auf skalare Werte abbildet. Mitte: Das odd-even Sortiernetzwerk sortiert die Skalare durch parallele bedingte Tauschoperationen so, dass alle Eingaben in ihre richtige Reihenfolge gebracht werden. Dieser Prozess wird differenzierbar gemacht, indem die Eingaben zu den bedingten Vertauschungsoperationen durch eine Verteilung (z.B. logistische oder Cauchy) gestört werden und der Erwartungswert der Ausgänge in geschlossener Form bestimmt wird. Rechts: Das Sortiernetz erzeugt eine differenzierbare Permutationsmatrix P, die dann mit der Permutationsmatrix Q der Grundwahrheit unter Verwendung der binären Cross-Entropy verglichen werden kann, um den Fehler zu ermitteln. Indem dieses Fehlersignal rückwärts durch das Sortiernetz propagiert wird, wird das CNN trainiert.

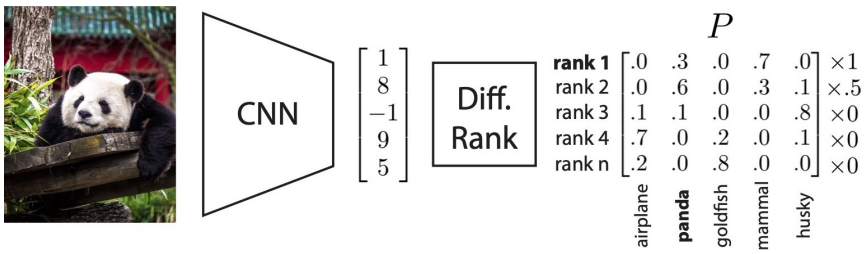

Abb. 5: Überblick über das Top-k-Klassifikations-Loss: Ein CNN sagt Bewertungen für ein Bild voraus, welche dann durch einen differenzierbaren Ranking-Algorithmus geordnet werden, welcher eine Wahrscheinlichkeitsverteilung für jeden Rang in der Matrix **P** zurückgibt. Im Beispiel verwenden wir 50% Top-1 und 50% Top-2 Komponenten, d.h. $P_K = [.5,.5,0,0,0]$. Die Gewichte für die verschiedenen Ränge können also über eine kumulative Summe berechnet werden und sind $[1,.5,0,0,0]$. Die entsprechend gewichtete Summe der Zeilen von **P** ergibt die Wahrscheinlichkeitsverteilung $p = [0,.6,0,.85,.05]$, die in der Cross-Entropy Verlustfunktion verwendet wird. p ist die Wahrscheinlichkeit für eine Klasse zu den Top-k zu gehören, wobei k aus P_K gezogen wird.

Differenzierbares Top-k. Aufbauend auf differenzierbaren Sortieralgorithmen führt die Dissertation differenzierbare Top-k Operatoren ein, d.h., Operatoren, welche die k größten Werte aus einer Liste oder einem Vektor identifizieren. Basierend hierauf wird eine Top-k-Klassifizierungs-Verlustfunktion eingeführt, mit welcher die Arbeit neue State-of-the-Art Resultate für öffentlich verfügbare Modelle auf dem ImageNet Datensatz [De09] erreicht hat [Pe22c]. Dies wurde auf der ICML 2022 veröffentlicht.

Differenzierbares Rendering. Im Bereich des differenzierbaren Renderns stellt die Dissertation einen neuen differenzierbaren Renderer vor: GenDR, der generalisierte differenzierbare Renderer [Pe22e], welcher auf der CVPR 2022 veröffentlicht wurde. GenDR generalisiert differenzierbare Renderer aus zwei Perspektiven. Einerseits unterstützt GenDR eine Vielzahl an Wahrscheinlichkeitsverteilungen für die Störungen, die den Renderer differenzierbar machen. Andererseits generalisiert GenDR die Aggregation von Wahrscheinlichkeiten über beliebige T-Conormen. Die Arbeit untersucht, wie sich die Auswahl an Verteilung und T-Conorm auf die Genauigkeit und Qualität der Gradienten auf verschiedenen Problemstellungen auswirkt und kommt zu dem Schluss, dass diese Auswahl entscheidend ist. Zudem wird festgestellt, dass für ähnliche Daten ähnliche Verteilungen gut funktionieren, und dass die uniforme Verteilung im Durchschnitt über verschiedene Klassen an Daten am besten funktioniert.

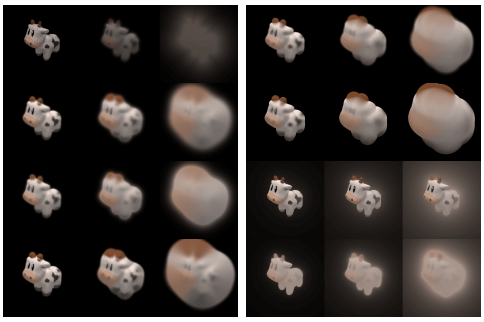

Abb. 6: Visueller Vergleich verschiedener Instanzen von GenDR. In jedem Bild steigt die Temperatur τ der Verteilung von links nach rechts an. Links: die logistische Verteilung und verschiedene T-Conormen für die Aggregation (von oben nach unten: $\perp^M, \perp^P, \perp_2^Y, \perp_{0.5}^A$). Rechts: erste zwei Zeilen nutzen die uniforme Verteilung mit $\perp_2^Y$ und $\perp_{0.5}^A$. Die letzten zwei Zeilen nutzen die Cauchy Verteilung mit $\perp^P$ und $\perp_2^Y$.

Weitere Themen. Über die hier diskutierten Themen hinaus behandelt die Dissertation auch differenzierbare Logik und differenzierbare Logik-Gatter-Netzwerke, sowie alternative Trainingsstrategien für Modelle mit integrierten Algorithmen. Differenzierbare Logik-Gatter-Netzwerke [Pe22b] erlauben das Training von Logik-Gatter-Netzwerke, welche, da Computer aus ihnen aufgebaut sind, besonders schnelle Inferenz erlauben, z.B. über eine Million Bilder pro Sekunde auf dem MNIST Datensatz (veröffentlicht auf der NeurIPS 2022). Das Kapitel über alternative Trainingsstrategien stellt eine Reihe neuartiger Methoden für das Training mit komplexen Verlustfunktionen (also Verlustfunktionen, die Algorithmen beinhalten) vor. Dies beinhaltet Optimierungsmethoden, die Informationen zweiter Ordnung betrachten, sowie Methoden für das Training mit nicht-differenzierbaren Black-box Algorithmen.

Literaturverzeichnis

[ALT16] Abernethy, Jacob; Lee, Chansoo; Tewari, Ambuj: Perturbation techniques in online learning and optimization. Perturbations, Optimization, and Statistics, 2016.

[Be20] Berthet, Quentin; Blondel, Mathieu; Teboul, Olivier; Cuturi, Marco; Vert, Jean-Philippe; Bach, Francis: Learning with Differentiable Perturbed Optimizers. In: Advances in Neural Information Processing Systems (NeurIPS). 2020.

[Ch19] Chen, Wenzheng; Gao, Jun; Ling, Huan; Smith, Edward J.; Lehtinen, Jaakko; Jacobson, Alec; Fidler, Sanja: Learning to Predict 3D Objects with an Interpolation-based Differentiable Renderer. In: Advances in Neural Information Processing Systems (NeurIPS). 2019.

[Co21] Cordonnier, Jean-Baptiste; Mahendran, Aravindh; Dosovitskiy, Alexey; Weissenborn, Dirk; Uszkoreit, Jakob; Unterthiner, Thomas: Differentiable Patch Selection for Image Recognition. In: Proc. International Conference on Computer Vision and Pattern Recognition (CVPR). 2021.

[CSL10] Chaudhuri, Swarat; Solar-Lezama, Armando: Smooth Interpretation. In: Proceedings of the 31st ACM SIGPLAN Conference on Programming Language Design and Implementation. 2010.

[CTV19] Cuturi, Marco; Teboul, Olivier; Vert, Jean-Philippe: Differentiable Ranking and Sorting using Optimal Transport. In: Advances in Neural Information Processing Systems (NeurIPS). 2019.

[De09] Deng, Jia; Dong, Wei; Socher, Richard; Li, Li-Jia; Li, Kai; Fei-Fei, Li: ImageNet: A Large-Scale Hierarchical Image Database. In: Proc. International Conference on Computer Vision and Pattern Recognition (CVPR). 2009.

[Fu80] Fukushima, Kunihiko: Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics, 36:193–202, 1980.

[Gr19] Grover, Aditya; Wang, Eric; Zweig, Aaron; Ermon, Stefano: Stochastic Optimization of Sorting Networks via Continuous Relaxations. In: International Conference on Learning Representations (ICLR). 2019.

[Ka20] Kato, Hiroharu; Beker, Deniz; Morariu, Mihai; Ando, Takahiro; Matsuoka, Toru; Kehl, Wadim; Gaidon, Adrien: Differentiable Rendering: A Survey. Computing Research Repository (CoRR) in arXiv, 2020.

[KB15] Kingma, Diederik; Ba, Jimmy: Adam: A Method for Stochastic Optimization. In: International Conference on Learning Representations (ICLR). 2015.

[Kn98] Knuth, Donald E.: The Art of Computer Programming, Volume 3: Sorting and Searching (2nd Ed.). Addison Wesley, 1998.

[KSH12] Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffrey E: ImageNet Classification with Deep Convolutional Neural Networks. In: Advances in Neural Information Processing Systems (NeurIPS). 2012.

[KWM16] Kirk, David B; Wen-Mei, W Hwu: Programming Massively Parallel Processors: A Hands-on Approach. Morgan Kaufmann, 2016.

[Le99] LeCun, Yann; Haffner, Patrick; Bottou, Léon; Bengio, Yoshua: Object Recognition with Gradient-Based Learning. In: Shape, contour and grouping in computer vision, S. 319–345. Springer, 1999.

[Li18] Li, Tzu-Mao; Aittala, Miika; Durand, Fredo; Lehtinen, Jaakko: Differentiable Monte Carlo Ray Tracing Through Edge Sampling. ACM Transactions on Graphics (Proc. SIGGRAPH Asia), 2018.

[Li19] Liu, Shichen; Li, Tianye; Chen, Weikai; Li, Hao: Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning. In: Proc. International Conference on Computer Vision (ICCV). 2019.

[Li21] Lidec, Quentin Le; Laptev, Ivan; Schmid, Cordelia; Carpentier, Justin: Differentiable Rendering with Perturbed Optimizers. In: Advances in Neural Information Processing Systems (NeurIPS). 2021.

[ML43] Menabrea, Luigi F; Lovelace, Ada A: Sketch of the Analytical Engine, invented by Charles Babbage, Esq., by LF Menabrea, of Turin, officer of the Military Engineers. Translated and with notes by AA, L. Taylor's Scientific Memoirs, 3:666–731, 1843.

[MP43] McCulloch, Warren S; Pitts, Walter: A logical calculus of the ideas immanent in nervous activity. The Bulletin of Mathematical Biophysics, 5(4):115–133, 1943.

[Ne69] Neugebauer, O.: The Exact Sciences in Antiquity. Acta historica scientiarum naturalium et medicinalium. Dover Publications, 1969.

[Ni19] Nimier-David, Merlin; Vicini, Delio; Zeltner, Tizian; Jakob, Wenzel: Mitsuba 2: A retargetable forward and inverse renderer. ACM Transactions on Graphics (TOG), 38(6):1–17, 2019.

[Pa19] Paszke, Adam; Gross, Sam; Massa, Francisco; Lerer, Adam; Bradbury, James; Chanan, Gregory; Killeen, Trevor; Lin, Zeming; Gimelshein, Natalia; Antiga, Luca; Desmaison, Alban; Kopf, Andreas; Yang, Edward; DeVito, Zachary; Raison, Martin; Tejani, Alykhan; Chilamkurthy, Sasank; Steiner, Benoit; Fang, Lu; Bai, Junjie; Chintala, Soumith: PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Advances in Neural Information Processing Systems (NeurIPS). 2019.

[Pe19] Petersen, Felix; Bermano, Amit H; Deussen, Oliver; Cohen-Or, Daniel: Pix2Vex: Image-to-Geometry Reconstruction using a Smooth Differentiable Renderer. Computing Research Repository (CoRR) in arXiv, 2019.

[PE20] Prillo, Sebastian; Eisenschlos, Julian: SoftSort: A continuous relaxation for the argsort operator. In: Proc. Machine Learning Research (PMLR), International Conference on Machine Learning (ICML). 2020.

[Pe21a] Petersen, Felix; Borgelt, Christian; Kuehne, Hilde; Deussen, Oliver: Differentiable Sorting Networks for Scalable Sorting and Ranking Supervision. In: Proc. Machine Learning Research (PMLR), International Conference on Machine Learning (ICML). 2021.

[Pe21b] Petersen, Felix; Borgelt, Christian; Kuehne, Hilde; Deussen, Oliver: Learning with Algorithmic Supervision via Continuous Relaxations. In: Advances in Neural Information Processing Systems (NeurIPS). 2021.

[Pe22a] Petersen, Felix: Learning with Differentiable Algorithms. Dissertation, Universität Konstanz, 2022.

[Pe22b] Petersen, Felix; Borgelt, Christian; Kuehne, Hilde; Deussen, Oliver: Deep Differentiable Logic Gate Networks. In: Advances in Neural Information Processing Systems (NeurIPS). 2022.

[Pe22c] Petersen, Felix; Borgelt, Christian; Kuehne, Hilde; Deussen, Oliver: Differentiable Top-k Classification Learning. In: Proc. Machine Learning Research (PMLR), International Conference on Machine Learning (ICML). 2022.

[Pe22d] Petersen, Felix; Borgelt, Christian; Kuehne, Hilde; Deussen, Oliver: Monotonic Differentiable Sorting Networks. In: International Conference on Learning Representations (ICLR). 2022.

[Pe22e] Petersen, Felix; Goldluecke, Bastian; Borgelt, Christian; Deussen, Oliver: GenDR: A Generalized Differentiable Renderer. In: Proc. International Conference on Computer Vision and Pattern Recognition (CVPR). 2022.

[Pe22f] Petersen, Felix; Goldluecke, Bastian; Deussen, Oliver; Kuehne, Hilde: Style Agnostic 3D Reconstruction via Adversarial Style Transfer. In: IEEE Winter Conference on Applications of Computer Vision (WACV). 2022.

[Re16] Redmon, Joseph; Divvala, Santosh; Girshick, Ross; Farhadi, Ali: You Only Look Once: Unified, Real-Time Object Detection. In: Proc. International Conference on Computer Vision and Pattern Recognition (CVPR). 2016.

[RHW86] Rumelhart, David E; Hinton, Geoffrey E; Williams, Ronald J: Learning representations by back-propagating errors. Nature, 323(6088):533–536, 1986.

[Se20] Senior, Andrew W.; Evans, Richard; Jumper, John; Kirkpatrick, James; Sifre, Laurent; Green, Tim; Qin, Chongli; Žídek, Augustin; Nelson, Alexander W. R.; Bridgland, Alex; Penedones, Hugo; Petersen, Stig; Simonyan, Kareand Crossan, Steve; Kohli, Pushmeet; Jones, David T.; Silver, David; Kavukcuoglu, Koray; Hassabis, Demis: Improved Protein Structure Prediction using Potentials from Deep Learning. Nature, 2020.

[Sh23] Shvetsova, Nina; Petersen, Felix; Kukleva, Anna; Schiele, Bernt; Kuehne, Hilde: Learning by Sorting: Self-supervised Learning with Group Ordering Constraints. Computing Research Repository (CoRR) in arXiv, 2023.

[Va17] Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N; Kaiser, Ł ukasz; Polosukhin, Illia: Attention is All you Need. In: Advances in Neural Information Processing Systems (NeurIPS). 2017.

[Zh21] Zhang, Cheng; Dong, Zhao; Doggett, Michael; Zhao, Shuang: Antithetic sampling for Monte Carlo differentiable rendering. ACM Transactions on Graphics (TOG), 40(4):1–12, 2021.

Felix Petersen wurde am 12. Oktober 1999 in Lübeck geboren. Das Bachelorstudium im Fach Informatik absolvierte Dr. Petersen an der Universität Konstanz im Jahr 2019 mit Auszeichnung. Am 22. August 2022 verteidigte Dr. Petersen seine Dissertation zum Thema “Learning with Differentiable Algorithms” an der Universität Konstanz mit der Note summa cum laude (0.0). Dr. Petersen hat auf den Top-Tier Konferenzen im Bereich maschinellen Lernens veröffentlicht, darunter NeurIPS (4x), ICML (2x), ICLR (2x) und CVPR (1x). Dr. Petersens Forschungsinteressen liegen im Bereich des maschinellen Lernens, mit Fokus auf differenzierbare Algorithmen, Optimierung, algorithmischer Fairness, Deep Learning und effizienten Berechnungsverfahren. Nach der Promotion begann Dr. Petersen einen Postdoc an der Stanford University.