

■ fakultät für informatik

Masterarbeit

Zeit-effizientes Training von Convolutional Neural Networks

Jessica Bühler
30. Dezember 2019

Gutachter:

Prof. Dr. Heinrich Müller
M.Sc. Matthias Fey

Lehrstuhl VII
Informatik
TU Dortmund

Inhaltsverzeichnis

1	Einleitung	1
1.1	Motivation und Hintergrund dieser Arbeit	1
1.2	Aufbau der Arbeit	1
2	Stand der Wissenschaft	3
2.1	Funktionsweise eines CNN	3
2.1.1	Faltung	3
2.2	Suchbegriffe	3
2.3	verwendete Datensets	3
2.4	Überblick über die gängigen Methoden	4
2.4.1	Verringerung der für Berechnungen nötige Zeit	5
2.4.2	Berechnung mit 16 Bit Dynamischen Festkommazahlen	6
2.5	Beschleunigung der Berechnung des Gradientenabstiegsverfahren	7
2.5.1	Accelerating CNN Training by Sparsifying Activation Gradients	7
2.5.2	Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks	7
2.5.3	Accelerating Deep Neural Network Training with Inconsistent Stochastic Gradient Descent	7
2.5.4	Accelerated CNN Training Through Gradient Approximation	7
2.6	Verfahren um weniger Trainingsdaten zu verwenden	8
2.6.1	Stochastisches Pooling	8
2.6.2	Lernen von Struktur und Stärke von CNNs	8
2.7	Strukturelle Veränderungen zur Beschleunigung des Trainings	9
2.7.1	Pruning um Trainingszeit zu minimieren	9
2.7.2	Net 2 Net	11
2.7.3	Kernel rescaling	11
2.7.4	Resource Aware Layer Replacement	11
2.8	Weitere Herangehensweisen	12
2.8.1	Tree CNN	12
2.8.2	Standardization Loss	12

2.8.3	Wavelet	12
3	Experimentelle Untersuchung der möglichen Strategien	13
3.1	Experimentales Setup	13
3.2	Überblick über die möglichen Strategien	13
3.2.1	Zahlenformate	13
3.2.2	Beschleunigung der Berechnung des Gradientenabstiegverfahren	14
3.2.3	Verfahren um weniger Trainingsdaten zu verwenden	14
3.2.4	Strukturelle Veränderungen	14
3.2.5	andere Herangehensweisen	14
3.2.6	Tensorflow vs. PyTorch	14
3.3	Durchführung der Experimente	14
3.3.1	Kombination von Net2Net mit PruneTrain	14
3.4	Evaluation der Ergebnisse	14
4	Konklusion	15
I	Additional information	17
	Abbildungsverzeichnis	19
	Algorithmenverzeichnis	21
	Quellcodeverzeichnis	23
	Literaturverzeichnis	25

Liste der noch zu erledigenden Punkte

■ Einleitung fertig schreiben – zum Schluss	1
■ Aufbau der Arbeit – erst bei fortgeschrittener Arbeit schreiben	1
■ Erkläre wie die Faltung im Kontext eines CNNs funktioniert	3
■ CNN Erklärung	3
■ verwendete Suchbegriffe	3
■ verwendete Datensets	3
■ kurzer Sectioneinleitungstext über die gängigen Methoden	4
■ Fehlt hier noch etwas?	5
■ cite	5
■ subnormale Zahlen	5
■ ref	6
■ ref	6
Abbildung: Schema	6

Mathematical Notation

Notation	Meaning
\mathbb{N}	Set of natural numbers $1, 2, 3, \dots$
\mathbb{R}	Set of real numbers
\mathbb{R}^d	d -dimensional space
$\mathcal{M} = \{m_1, \dots, m_N\}$	Set \mathcal{M} of N elements m_i
\mathbf{p}	Vector
\mathbf{p}_i	Element i of the vector
$\mathbf{v}_i^{(j)}$	Element i of the vector j
\mathbf{A}	Matrix

1 Einleitung

1.1 Motivation und Hintergrund dieser Arbeit

Gegeben: Mehrere Datensätze aus Bildern, die in verschiedene vorgegebene Klassen klassifiziert werden sollen.

Gesucht: Effiziente Strategie um möglichst zeitsparend eine möglichst gute Klassifikationsleistung zu bekommen

Also ein Optimierungsproblem auf verschiedenen Strategien zum Trainieren von CNNs mit zwei zu Optimierenden Größen:

- Zeit effizientes Training
- Gute Klassifikationsleistung

Gesucht ist also eine Pareto-Front

Einleitung fertig schreiben – zum Schluss

1.2 Aufbau der Arbeit

Aufbau der Arbeit – erst bei fortgeschrittener Arbeit schreiben

2 Stand der Wissenschaft

2.1 Funktionsweise eines CNN

Die Quelle für dieses Unterkapitel ist soweit nicht anders vermerkt ein Buch über „Deep Learning“ [GBC16]. CNNs sind neuronale Netze, die in mindestens einer ihrer Schichten die Faltung anstelle der allgemeinen Matrixmultiplikation verwenden. Als Eingabedaten sind für das CNN gridförmige Daten geeignet. Da Bilder im Computer in der Regel gridförmig abgespeichert werden, ist die Bildverarbeitung eine große Anwendungsdomäne von CNNs. Im Bereich der Bildverarbeitung werden CNNs beispielsweise für diese Aufgaben verwendet:

- Bildklassifikation
- Bildsegmentierung
- Bildmanipulation

2.1.1 Faltung

Erkläre wie die Faltung im Kontext eines CNNs funktioniert

CNN Erklärung

2.2 Suchbegriffe

verwendete Suchbegriffe

2.3 verwendete Datensets

verwendete Datensets

2.4 Überblick über die gängigen Methoden

kurzer Sectioneinleitungstext über die gängigen Methoden

2.4.1 Verringerung der für Berechnungen nötige Zeit

Die Zeit, die ein Convolutional Layer braucht um berechnet zu werden hängt ab von:

- der Filtergrösse
- der Bildgrösse
- dem verwendeten Zahlenformat

Fehlt hier
noch etwas

Beim Verändern der Filter- oder der Bildgrösse, um Trainingszeit zu sparen, verändert sich auch die Erkennungsleistung. Dies ist beim Verändern des verwendeten Zahlenformats nicht unbedingt gegeben. Standardformat ist eine 32 Bit Gleitkommazahl. Die einfachste Methode hier Trainingszeit zu sparen ist das Halbieren der Bitanzahl auf 16 Bit. Eine weitere Methode ist das Benutzen von 16 Bit Dynamischen Festkommazahlen. Die beiden alternativen Methoden haben unterschiedliche Anforderungen an die Ausführungsplattform. Diese Anforderungen und die Besonderheiten der beiden Verfahren werden in den folgenden zwei Unterkapiteln näher beleuchtet.

cite

Berechnung mit 16 Bit Gleitkomma [?] Die 16 Bit Gleitkommazahl unterscheidet sich nicht nur in der Länge von der 32 Bit Zahl sondern aus der unterschiedlichen Länge erwachsen Unterschiede in den darstellbaren Zahlen. In Tabelle 2.1 sind diese Unterschiede dargestellt.

Tabelle 2.1: Darstellbare Zahlen von 16 und 32 Bit

	16 Bit	32 Bit
kleinste darstellbare positive Zahl	$0.61 \cdot 10^{-4}$	$1.1755 \cdot 10^{-38}$
grösste darstellbare positive Ganzzahl	65504	$3.403 \cdot 10^{38}$
minimal subnormale Zahl	$2^{-24} \approx 5.96 \cdot 10^{-8}$	2^{-149}

Subnormale Zahlen ergeben sich, wenn der Exponent 0 ist und

subnormale Zahlen

Durch diese Unterschiede im Umfang der darstellbaren Zahlen ergibt sich ein direkter Unterschied im Training eines CNNs. Durch den Wechsel auf 16 Bit ist ein bestimmter Teil der Gradienten gleich Null.

Diese Nachteile von 16 Bit Gleitkommazahlen können durch drei Techniken abgemildert oder sogar komplett aufgehoben werden:

- 32 Bit Mastergewichte und Updates
- Sklaierung der Loss-Funktion
- Arithmetische Präzision

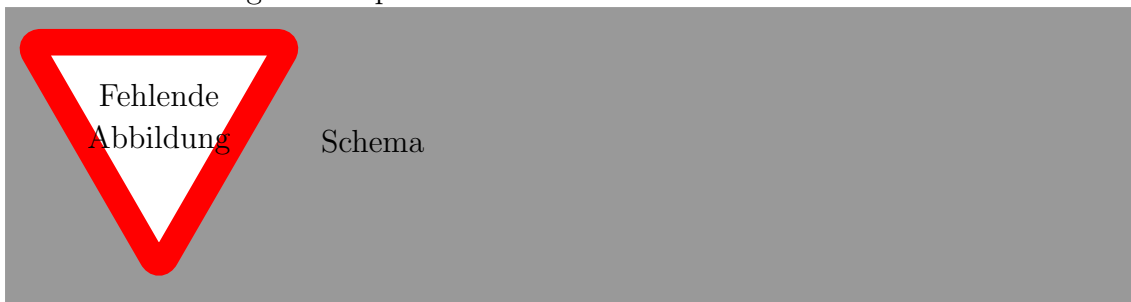
Diese drei Techniken werden in den drei folgenden Unterkpaiteln behandelt.

32 Bit Mastergewichte und Updates Beim Trainieren von neuronalen Netzwerken mit 16 Bit Gleitkommazahlen werden die Gewichte, Aktivierungen und Gradienten im 16 Bit Format gespeichert. Die Speicherung der Gewichte als 32 Bit Mastergewichte hat zwei mögliche Erklärungen, die aber nicht immer zutreffen müssen.

Um nach einem Forward Durchlauf des Netzes die Gewichte abzudaten wird ein Gradientenabstiegsverfahren benutzt. Hierbei werden die Gradienten der Gewichte berechnet. Um für die Funktion, die das CNN approximiert einen besseren Approximationserfolg zu erlangen wird dann dieser Gradient mit der Lernrate multipliziert. Wird dieses Produkt in 16 Bit abgespeichert, so ist in viele Fällen das Produkt der beiden Zahlen gleich Null. Dies liegt an der Taqtsache, dass wie in Tabelle zu sehen ist die kleinste darstellbare Zahl in 16 Bit wesentlich grösser ist als in 32 Bit.

Der zweite Grund wieso man Mastergewichte brauchen könnte ist die Tatsache, dass bei grossen Gewichten die Länge der Mantisse nicht ausreicht, um sowohl das Gewicht als auch das zu addierende Update zu speichern.

Aus den beiden Gründen wird das in Abbildung gezeigte Schema zum Trainieren einer Schicht mit gemischt präzisen Gleitkommazahlen benutzt.



Sklaierung der Loss-Funktion

Arithmetische Präzision

2.4.2 Berechnung mit 16 Bit Dynamischen Festkommazahlen

Quelle: [DMM⁺18]

2.5 Beschleunigung der Berechnung des Gradientenabstiegsverfahren

Bei der Beschleunigung der Berechnung des Gradientenabstiegsverfahren gibt es vier verschiedene publizierte Herangehensweisen:

- Accelerating CNN Training by Sparsifying Activation Gradients
- Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks
- Accelerating Deep Neural Network Training with Inconsistent Stochastic Gradient Descent
- Accelerated CNN Training Through Gradient Approximation

2.5.1 Accelerating CNN Training by Sparsifying Activation Gradients

Funktioniert nur auf Toy-Benchmarks

2.5.2 Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks

2.5.3 Accelerating Deep Neural Network Training with Inconsistent Stochastic Gradient Descent

2.5.4 Accelerated CNN Training Through Gradient Approximation

2.6 Verfahren um weniger Trainingsdaten zu verwenden

2.6.1 Stochastisches Pooling

2.6.2 Lernen von Struktur und Stärke von CNNs

2.7 Strukturelle Veränderungen zur Beschleunigung des Trainings

2.7.1 Pruning um Trainingszeit zu minimieren

Pruning ist eine Technik, die entwickelt wurde, um die Inferenzzeit eines neuronalen Netzwerks zu reduzieren. Das Pruningverfahren wird auf das bereits trainierte Netz angewendet. Dabei wird entschieden, welche Gewichte nur einen minimalen Effekt auf das Klassifikationsergebnis haben um diese zu entfernen.

Aktueller Gegenstand der Forschung ist hier die Frage, ob diese kleineren Netzwerke nicht bereits ab Epoche Null trainiert werden können, um so Trainingszeit zu sparen. Dieser Ansatz wurde in verschiedenen Veröffentlichungen untersucht:

- Prune Train
- The Lottery Ticket Hypothesis

Zunächst werden die einzelnen Verfahren erläutert, um sie danach miteinander zu vergleichen.

Prune Train

Prune Train fügt einen Normalisierungsterm zur Loss-Funktion des Netzwerkes hinzu. Dies geschieht, damit der Optimierungsprozess dazu gezwungen wird möglichst kleine Gewichte zu wählen. Durch diesen Prozess wird aus dem dense Netz ein sparse Netz. Dieses sparse Netz sorgt allerdings noch nicht für weniger Zeitbedarf einer Trainingsepoche, da für ein Sparse aufwändige Datenindextechniken notwendig sind. Daher wird bei diesem Verfahren das Netz rekonfiguriert um das Modell kleiner und die Struktur wieder dense zu machen. Dabei hat Prune Train drei zentrale Optimierungsverfahren:

- eine systematische Methode zur Berechnung des group lasso Regularisierung Sanktions Koeffizienten beim Beginn des Trainings.
- Kanal union, ein Speicheraufruf kosteneffizientes und Index-freies Kanal Pruning Verfahren für moderne CNNs mit Kurzschlussverbindungen.
- Ein dynamische Mini-Batch Adjustment, dass die Größe des Mini-Batch anpasst. Dies geschieht durch beobachten des Speicherkapazitätgebrauchs einer Trainingsiteration nach jeder Pruning reconfiguration.

Der group lasso Regularisierung Sanktions Koeffizienten ist ein Hyperparameter, der einen Trade-off zwischen der Modellgröße und der Accuracy bildet. Voherige Arbeiten suchen nach einem geeignetem Sanktionsmaß, was das Einbeziehen des Prunings vom Anfang des Trainings sehr teuer macht. Unser Mechanismus kontrolliert die Group lasso Regularisierungstärke und erreicht eine hohe Modellpruningrate mit nur einem kleinen Einfluss auf die Accuracy bei nur einem Trainingsdurchlauf. Kurzschlussverbindungen werden in modernen CNNs häufig genutzt. Pruning aller genullten Kanäle solcher CNNs brauchen regelmässige Tensor Umordnung um die Kanalindizes zwischen den Schichten zu matchen. Dies vermindert die Performance. Diese Umordnung wird durch den Channel Union Algorithmus vermieden. Daher folgt eine 1.9 fache Beschleunigung des Convolutional Layetrs.

Dynmaisches Mini Batch Adjustment kompensiert die verminderte Datenparallelität aufgrund des kleineren geprunten Modells durch Erhöhung der Mini-Batch Größe. Dies sorgt sowohl für bessere Ausnutzung der Hardware ressourcen als auch zur Reduzierung des KOMmunikation overheads durch eine Verminderte Modell Update Frequenz. Beim Erhöhen der Mini-Batch Größe wird auch die Lernrate mit demselben Verhältnis erhöht, um die Accuracy nicht zu verändern.

$$W_{min} \left(\frac{1}{N} \sum_{i=1}^N l(y_i, f(x_i, W)) + \sum_{g=1}^G \lambda_g \cdot \|W_g\|_2 \right)$$

- $\frac{1}{N} \sum_{i=1}^N l(y_i, f(x_i, W))$ Standard-Kreuzentropie
- $\sum_{g=1}^G \lambda_g \cdot \|W_g\|_2$ group lasso Regulierungsterm
- $f(x_i)$ Vorhersage des Netzwerks auf Eingabe x_i
- W Gewichte
- l Verlustfunktion der Klassifikation und Grundwahrheit y_i
- N Minibatchgröße
- G Zahl von Gruppen
- λ Verdünnungskoeffizient

$$\lambda \cdot \sum_{l=1}^L \left(\sum_{c_l=1}^{C_l} \|W_{c_l, :, :, :}\|_2 + \sum_{k_l=1}^{K_l} \|W_{:, k_l, :, :}\|_2 \right)$$

Design eines speziellen Groupo Lasso Regulierers, der Gewichte jedes Kanals (Input oder Output) und jeder Schicht gruppiert. λ wird als einziger globaler Regularisierungsfaktor gewählt, da so der Fokus auf dem Vermindern der Rechenzeit liegt und nicht auf der Modellgröße. Dies hat zur Folge, dass vorallem große Features verdünnt werden, was zu einer größeren Verminderung der Rechenleistung führt.

Um die Lasso Group Regularisierung vom Anfang des Trainings zu benutzen sollteder Koeffizient λ sinnvoll gewählt werden. Dies sorgt für eine hohe Vorhersageaccuracy und einer hohen Pruning Rate. Um zeitintensives Hyperparametertuning zu vermeiden wird hier eine neue Methode eingeführt:

$$LPR = \frac{\lambda \sum_g ||W_{g,:}||}{l(y_i, f(x_i, W)) + \lambda \sum_g ||W_{g,:}||}$$

Berechnet wird dies durch setzen von zufälligen Werten, mit denen die Gewichte initialisiert werden. LPR wird einmal berechnet und dann bis zum Ende weiter benutzt.

Nach jedem solchen Intervall werden Input und Outputkanäle die 0 sind gepruned. Um ein Missverhältnis zwischen den Dimensionen zu verhindern wird nur die Verbindung von 2 verdünnten Kanälen von 2 aufeinanderfolgenden Schichten gepruned. Alle Trainingsvariablen bleiben gleich. Das Reconfigurationsintervall ist ist der einzige zusätzliche Hyperparameter. Zu gross gewählt würde der Intervallparameter zu wenig Zeitverbesserung bringen. Zu klein gewählt könnte er die Lernqualität beeinflussen. 4 Matriken zur Evaluierung: Training und Inference FLOPs, gemessenen Trainingszeit, und Validierungsaccuracy.

The Lottery Ticket Hypothesis

2.7.2 Net 2 Net

2.7.3 Kernel rescaling

2.7.4 Resource Aware Layer Replacement

2.8 Weitere Herangehensweisen

2.8.1 Tree CNN

2.8.2 Standardization Loss

2.8.3 Wavelet

3 Experimentelle Untersuchung der möglichen Strategien

3.1 Experimentales Setup

Welche Hardware und damit zusammenhängend welche Versionen der dazugehörenden Software sind vorhanden — Daraus erwächst die Auswahl welche Strategien überhaupt möglich sind

3.2 Überblick über die möglichen Strategien

Welchen Strategien aus Kapitel 2 sind überhaupt durchführbar und welche sind kombinierbar? Hier werden nur die Strategien aufgeführt, welche überhaupt auf vernünftig grossen Datensätzen funktionieren und von der Technik her möglich sind. Die Strategien sind aufgeteilt in Unterkapitel.

Alle möglichen Kombinationen von Strategien sind zuviele. Daher sinnvolle Vorauswahl treffen. Bei mehreren gleichartigen/ konkurrierenden Ansätze direkter Vergleich und dann den besten auswählen.

3.2.1 Zahlenformate

- FP16 bereits probiert
- DFP 16 without Swalp
- DFP 16 with Swalp

3.2.2 Beschleunigung der Berechnung des Gradientenabstiegverfahren

3.2.3 Verfahren um weniger Trainingsdaten zu verwenden

3.2.4 Strukturelle Veränderungen

3.2.5 andere Herangehensweisen

3.2.6 Tensorflow vs. PyTorch

3.3 Durchführung der Experimente

3.3.1 Kombination von Net2Net mit PruneTrain

Jedes Bildklassifizierungsproblem hat

3.4 Evaluation der Ergebnisse

4 Konklusion

Teil I

Additional information

Abbildungsverzeichnis

Algorithmenverzeichnis

Quellcodeverzeichnis

Literaturverzeichnis

- [DMM⁺18] Dipankar Das, Naveen Mellempudi, Dheevatsa Mudigere, Dhiraj D. Kalamkar, Sasikanth Avancha, Kunal Banerjee, Srinivas Sridharan, Karthik Vaidyanathan, Bharat Kaul, Evangelos Georganas, Alexander Heinecke, Pradeep Dubey, Jesús Corbal, Nikita Shustrov, Roman Dubtsov, Evarist Fomenko, and Vadim O. Pirogov. Mixed precision training of convolutional neural networks using integer operations. *CoRR*, abs/1802.00930, 2018.
- [GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.