Optimierer in Convolutional Neural Networks

Martina Brüning
s0540636

Betreuer: Patrick Baumann, M. Sc.

15.09.2021

Hochschule für Technik und Wirtschaft Berlin

University of Applied Sciences

Inhalt

- Einleitung
- Projektüberblick
- Grundlagen
  - Convolutional Operation
  - Kreuzentropie
  - Stochastic Gradient Descent (SGD)
  - Adaptive Moment Estimation (Adam)
  - Layer-wise Adaptive Moments optimizer for Batch training (LAMB)
- Zusammenfassung
- Quellen

Einleitung

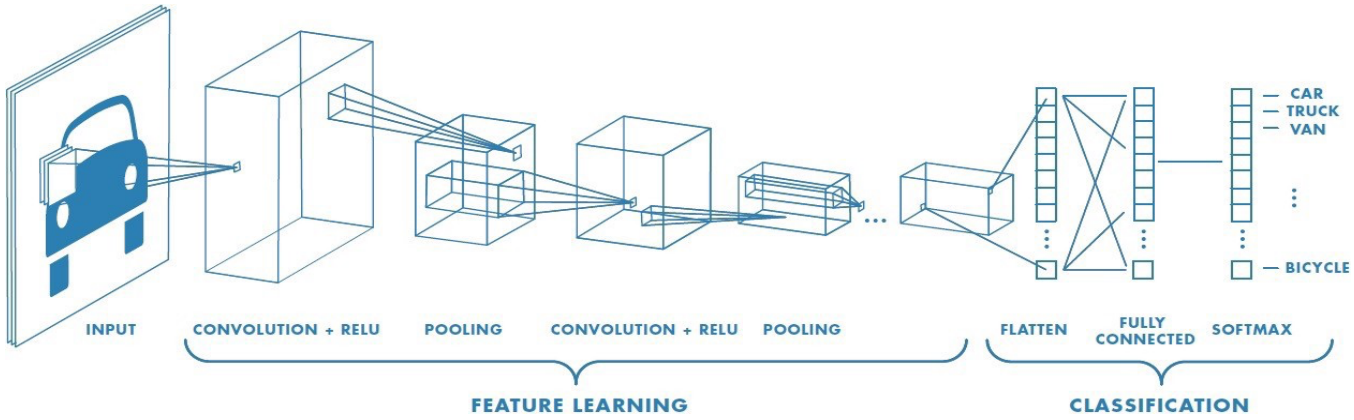
- Datenmengen steigen
  - Wetterbeobachtung
  - Medizin
  - Straßenverkehr
  - Onlinehandel
- Effiziente Verarbeitung der Daten ist notwendig

Projektüberblick

- Vergleich dreier gängiger Optimierer.
- CIFAR-10 dient als Bilddatenbank. (60.000 Bilder)
- Beobachtung der Leistungsänderung durch erhöhen der Stapelgröße.
- Erste Versuchsreihen mit 20 Durchläufen und 64 Bildern als Stapelgröße. (1.280 Bilder)
- Zweite Versuchsreihen mit 20 Durchläufen und 512 Bildern als Stapelgröße. (10.240 Bilder)

Grundlagen

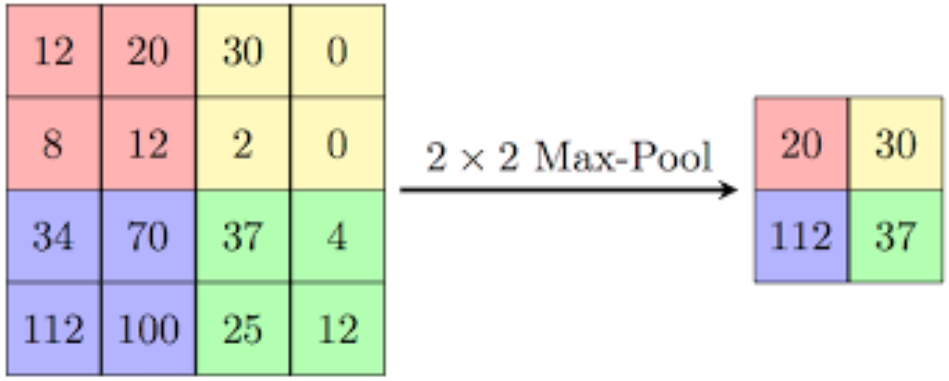
Convolutional Operation



- Mathematische Faltungsoperation
- Über das eingehende Bild iteriert ein Filter. Die resultierende Merkmalskarte bildet die erkannten Markmale ab.

Grundlagen

Convolutional Operation



- pooling operation verwirft überflüssige Bildinformationen
- Neue Bilddimension ist dann das Eingabebild für die folgende Schicht.

Grundlagen

Convolutional Operation

$$\dim\big(pooling(image)\big) = \left(\left[\frac{n_H + 2p - f}{s} + 1\right], \left[\frac{n_W + 2p - f}{s} + 1\right], n_c\right)$$

- neue Bildhöhe
- Neue Bildbreite
- Farbkanäle
- neue Bilddimesion

Legende: n_H = Bildhöhe, n_w = Bildbreite, p = Poolinggröße, f = Filtergröße, s = Schrittlänge, n_c = Farbkanäle

Grundlagen

Kreuzentropie

- Berechnet Genauigkeit zweier Wahrscheinlichkeiten
- Liefert eine Komponente für Optimierer
- Klassifikation in $[0\ ,1]$
- x = Eingabewert
- y = wahrer Wert

Grundlagen

Kreuzentropie

$$\frac{\partial}{\partial\theta} J\theta = -\frac{1}{N}\sum_{i=1}^{N}[y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)})\log(1 - h_\theta(x^{(i)})]$$

- Erster Term berechnen wenn $y = 1$.
- Zweiter Term berechnen wenn $y = 0$.
- Summe durch Anzahl Trainingsbeispiele dividieren.
- Verlust (Entropie)

Legende: N = Datenpunkte, y = wahrer Wert, x = errechneter Wert, h_θ = Modell, $\frac{\partial}{\partial\theta} J\theta$ = Gradient

Grundlagen

SGD

- Stochastic Gradient Descent
- Grundlage für Trainingsalgorithmen
- Einfach und effektiv
- Aktualisiert Gradienten nach jedem Stapeldurchlauf
- Effizienter Optimierer

Grundlagen

SGD

$$w_{k+1} = \beta * w_k - \alpha_k * \nabla f_k(w_k)$$

- Momentum * aktuelle Position
- Schrittlänge
- Resultat aus der Verlustfunktion
- Neue Position

Legende: w = Position, β = Momentum, α = Schrittlänge, ∇f = Gradient aus Kreuzentropie

Grundlagen

Adam

- Adaptive Moment Estimation
- Kombiniert AdaGrad und RMSProp
- Adpative Lernrate durch Einführung zweier Momenti m, v
  - m dient zur Berechnung des Durchnitts der Schrittgröße.
  - v dient der Berechnung der Varianz des Gradienten.
- β_1, β_2 konstante Verlustraten für Momentschätzung

Grundlagen

Adam

Schätzungen mithilfe der Verlustraten berechnen:

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * g_t$$

$$v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2$$

Korrektur der verzerrten Schätzungen:

$$\widehat{m_t} = \frac{m_t}{(1 - \beta_1^t)}$$

$$\widehat{v_t} = \frac{v_t}{(1 - \beta_2^t)}$$

Legende: m_t = 1. Schrittweitendurchschnitt, v_t = 2. Gradientenvarianz, β_1, β_2 = konstante Verlustraten, $\widehat{m}, \widehat{v}$ = korrigierte Werte, g = Gradient.

Adam

$$\theta_{t+1} = \theta_t - \alpha_t \frac{\widehat{m_t}}{\left(\sqrt{\widehat{v_t}} + \varepsilon\right)}$$

- Aktuelle Position
- Adaptive Schrittlänge
- Verhältnis der korrigierten Momenti
- Neue Position

Legende: $\widehat{m}$,$\widehat{v}$ = korrigierte Werte, α = Schrittweite, θ = Gradientenposition, ϵ = kleine Konstante zur numerischen Stabilität.

Grundlagen

LAMB

- Layer-wise Adaptive Moments optimizer for Batch training
- Erweiterung des Adam Algorithmus
- Adaptive Lernrate wird schichtweise berechnet
- Einführung eines Vertrauensverhältnisses (engl. trust ratio)
  - Korrigiert zu große (instabil) oder zu kleine (konvergiert langsam) Unterschieden von Gradient und Gewichtung.

Grundlagen

LAMB

Schätzungen mithilfe der Verlustraten berechnen:

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * g_t$$

$$v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2$$

Korrektur der verzerrten Schätzungen:

$$\widehat{m_t} = \frac{m_t}{(1 - \beta_1^t)}$$

$$\widehat{v_t} = \frac{v_t}{(1 - \beta_2^t)}$$

Legende: m_t = 1. Schrittweitendurchschnitt, v_t = 2. Gradientenvarianz, β_1, β_2 = konstante Verlustraten, $\widehat{m}, \widehat{v}$ = korrigierte Werte, g = Gradient.

Grundlagen

LAMB

Vertrauensverhätniskomponente:

Gewichtungen normalisieren: $r_1 = \phi(||w_{t-1||}^l)$

Elemntweiser Gewichtsverlust: $r_2 = \left| \frac{\widehat{m_t^l}}{\widehat{v_t^l} + \varepsilon} + \lambda * w_{t-1}^l \right|$

Vertrauensverhältnis: $r_t^l = \frac{r_1}{r_2}$

Legende: $\widehat{m}$,$\widehat{v}$ = korrigierte Werte, g = Gradient, w = Gradientenposition, r = Vertrauensverhätnis, λ = Gewichtsverlustskonstanten, ϵ = kleine Konstante zur numerischen Stabilität.

Grundlagen

LAMB

$$w_t^l = w_{t-1}^l - \eta * r_t^l * \left(\frac{\widehat{m_t^l}}{\widehat{v_t^l} + \varepsilon} + \lambda * w_{t-1}^l\right)$$

- neue Position
- aktuelle Position
- adaptive Schrittlänge
- Vertrauensverhältnis
- elementweise Gewichtsverlustsberechnung

Legende:
$\widehat{m}, \widehat{v}$ = korrigierte Werte, η = Schrittweite, w = Gradientenposition, r = Vertrauensverhätnis, λ = Gewichtsverlustskonstanten, ϵ = kleine Konstante zur numerischen Stabilität.

Zusammen-fassung

1. Testreihe mit 20 Epochen, 64 Bilder pro Stapel

	Trainingsverlust	Validierungsverlust	Generalisierungsverlust	Genauigkeit
SGD	0,007091	0,192426	1:27	79,19%
Adam	0,004356	0,217715	1:49	79,58%
LAMB	0,006589	0,203922	1:30	78,36%

2. Testreihe mit 20 Epochen und 512 Bilder pro Stapel

	Trainingsverlust	Validierungsverlust	Generalisierungsverlust	Genauigkeit
SGD	0,007091	0,192426	1:21	65,38%
Adam	0,004356	0,217715	1:32	78,91%
LAMB	0,006589	0,203922	1:21	73,23%

Quellen

- Saha, Sumit. 2018. *towards data science.* 15. 12. Zugriff am 08. 07 2021. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.
- WikiMedia Foundation. 2018. computersciencewiki. 27. 02. Zugriff am 02.09.2021. https://computersciencewiki.org/index.php/Max-pooling_/_Pooling
- Mebsout, Ismail. Zugriff am 06.09.2021. *towards data science.* Zugriff am 2021. 08 17. https://towardsdatascience.com/convolutional-neural-networks-mathematics-1beb3e6447c0.
- Baumann, Patrick. 2021. *HTW moodle.* Zugriff am 21. 08 2021. https://moodle.htw-berlin.de/pluginfile.php/1163173/mod_resource/content/3/03_Logistic_Regression.pdf.
- Bottou, L´eon, Frank E. Curtis, und Jorge Nocedal. 2018. *arxiv.* 08. 02. Zugriff am 08. 09 2021. https://arxiv.org/pdf/1606.04838.pdf.
- You, Yang, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, und Cho-Jui Hsieh. 2020. *arxiv.* 03. 01. Zugriff am 20. 08 2021. https://arxiv.org/pdf/1904.00962.pdf.

Quellen

- Science, NUS Department of Computer, Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, et al. 2020. *youtube.com.* 22. 09. Zugriff am 24. 08 2021. https://www.youtube.com/watch?v=kwEBP-Wbtdc&t=416s.
- Yiding Jiang, Dilip Krishnan, Hossein Mobahi, Samy Bengio. 2019 arxiv. 12.06. Zugriff am 13.09.2021. https://arxiv.org/pdf/1810.00113.pdf