Deep Learning - Lernen von vielschichtigen neuronalen Netzen

Kjartan Ferstl

DIPLOMARBEIT

eingereicht am

Fachhochschul-Masterstudiengang

Information Engineering und -Management

in Hagenberg

im Juni 2014

Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die aus anderen Quellen entnommenen Stellen als solche gekennzeichnet habe.

Hagenberg, am 12. Juni 2014

Kjartan Ferstl

Inhaltsverzeichnis

Kurzfassung

Deep Learning-Algorithmen biete eine Möglichkeit vielschichtige Neuronale Netze zu trainieren. Solche Netze kommen in aktuellen Anwendungen zu Erkennung von Objekten in Bildern, Text und Inhalt in Sprache und vielen anderen Gebieten zum Einsatz. Forschungen an den Algorithmen ermöglichen es, die sehr rechenintensiven Algorithmen zu vereinfachen und in großem Stil auf parallelisierten Computersystemen auszuführen. So ist es in einigen Gebieten bereits möglich mit neuronalen Netzen bessere Ergebnisse als mit optimierten Mathematischen verfahren zu erzielen. In folgendem werden die wesentlichen Algorithmen und Strukturen von neuronalen Netzen aufgearbeitet und die Schwierigkeiten und Lösungsansätze aufgezeigt. Außerdem werden einige aktuelle Anwendungen aus der Bild- und Sprachanalyse betrachtet.

Abstract

Deep learning algorithms enable it to train multi player neural networks. Such networks are used to recognize objects in images, to extract text and content in speach and for many other applications. Research in this area made it possible to reduce the complexity of such algorithms and to compute them in parallel on large scale computer networks. Due to these optimization, it is already possible, to produce better results with an neural network than with existing optimized mathematical algorithms for some applications. This paper is concerned about the major algorithms and structures used for neural networks, the problems they come with and how they can be solved. Moreover some applicactions from object recognition in images and voice analysis are examined.

Kapitel 1

Einleitung

1.1 Motivation

Komplexe neuronale Netze können bei einigen Problemstellungen, besonders in der Bild- und Spracherkennung ein sehr mächtiges Mittel zur Problemlösung sein. Das Lernen solcher Netze bringt Schwierigkeiten mit sich die in den vergangenen Jahrzenten schlecht oder gar nicht gelöst werden konnten.

Aufgrund der Weiterentwicklung von PC-Hardware und der Verwendung der Grafik-Recheneinheit (GPU), sind seit den 00er-Jahren bereits bemerkenswerte Ergebnisse möglich. Momentan arbeiten sogar Chiphersteller an preiswerten, dedizierten Chips mit trainierbaren neuronalen Netzen. Qualcomm wird den ersten kommerziellen Chip mit einem integrierten neuronalen Netz noch dieses Jahr (2014) veröffentlichen. Neuronale Netze sind immer nur so gut, wie sie trainiert werden, eines der wesentlichsten Themen im weiteren Fortschritt von neuronalen Netzen sind daher die Algorithmen zum Trainieren.

Aus der eingeleiteten Motivation ergeben sich daher folgende Kernaufgaben für die Seminararbeit:

- Einführung in die Problematik von Deep Learning mit kurzem Blick auf die bisherige Geschichte
- Analyse der Hürden im Deep Learning
- Analyse aktueller Deep Learning-Algorithmen

1.2 Ziele und Aufgaben

Folgende zentrale Fragestellungen sollen in der Seminararbeit beantwortet werden:

- Was ist Deep Learning?
- Welche Mustertypen können erkannt werden und welche Anwendungen sind möglich
- Wie sind die gelernten Modelle konkret definiert?
- Welche Verfahren werden verwendet um vielschichtige Modelle zu lernen?
- Was ist aktuell mit Deep Learning in Verbindung mit neuronalen Netzen bei der Mustererkennung in Bildern möglich?

Ziel dieser Arbeit ist es, dem interessierten Leser einen Einstieg in Themen deep learning, mit besonderem Fokus auf neuronale Netze zu vermitteln. Aufbauend auf diesen Erkenntnissen soll gezeigt werden, welche wesentlichen Algorithmen bis heute entwickelt wurden und welche technischen Problemstellungen mit diesen Algorithmen gelöst werden können oder sogar bereits heute damit gelöst werden. Ein wesentlicher Fokus soll dabei auf die Mustererkennung in Bildern gerichtet werden.

Kapitel 2

Grundlegendes

2.1 Definition

Deep Learning ist ein Gebiet aus dem maschinellen Lernen, das sich mit dem Trainieren von nicht linearen Modellen und hier meist mit vielschichtigen neuronalen Netzen befasst.

Deep Learning wird zum Beispiel zur Klassifizierung von Daten oder zur Extraktion von Merkmalen aus Daten eingesetzt.

2.2 Entstehung

Die Wurzeln des Deep Learnings gehen zurück in die 1950er Jahre, als Frank Rosenblatt eine Maschine namens Perceptron Blank2008 baute. Diese Maschine war in der Lage einige einfache Figuren wie Quadrate und Dreiecke zu erkennen. Für die damalige Zeit war das eine herausragende Maschine und regte den Gedanken, Maschinen zu bauen die Menschen imitieren können weiter an.

Knapp 20 Jahre später schrieb Marvin Minsky ein Buch das die Limitierungen der Maschine aufzeigte und einige scheinbar fundamentale Probleme aufwarf. Diese Erscheinung ließ die neuronalen Netze wieder in den Hintergrund rücken. In den 1980er Jahren hat, der heute renomierte Wissenschaftler in diesem Bereich, eine Maschine gebaut die bereits eine versteckte Schicht besaß und somit in der Lage war komplexere Aufgaben zu lösen. In dieser Zeit entstand auch der erste Backpropagation Algorithmus. Das trainieren dieser Maschine war sehr aufwendig und so verschwand auch sie bald wieder von der Bildfläche.

ref Werbos, Amari?, Parker, LeCun, Rumelhart, ..)

Erst mit einer Entdeckung 2006, erneut durch Geoff Hinton, die den Backpropagation Algorithmus erheblich vereinfachte, haben neuronale Netze mit

ref

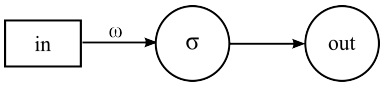


Abbildung 2.1: Neuron

mehreren versteckten Schichten wieder an Bedeutung gewonnen.

Heute stellen führende Unternehmen immer häufiger Teams rund um neuronale Netze und das Trainieren dieser zusammen um sich wirtschaftliche Vorteile zu sichern. Erste Erfolge sieht man an Microsofts Spracherkennung Cortana, Siri von Apple, Google Now oder der Bildersuche von Google, die mit Bildern ähnlichen Bildern sucht.

ref google, apple

ref

2.3 Hürden

Die aus der Vergangenheit bekannten Probleme beim Fortschritt neuronaler Netze mit vielen versteckten Schichten strecken sind zum größten Teil bis heute durch. Aktuelle Algorithmen aus dem Deep Learning tendieren zu Überanpassung, Unteranpassung, lokalen Minima oder kämpfen einfach mit der nicht vorhandenen Rechenleistung.

Grundsätzlich kann man für heute verwendete Algorithmen sagen, dass die Performance von neuronalen Netzen sehr stark von der Menge der Trainingsdaten und weniger von der Auswahl der Algorithmen abhängt. Nicht zuletzt hat die Verfügbarkeit von immensen Datenmengen von Unternehmen und aus dem Internet einen sehr positiven Einfluss auf das Interesse und die Entwicklungen im Bereich der Deep Learning Algorithmen.

2.4 Neuronale netze

Neuronale Netze sind Strukturen aus der Technik, die dem Nervensystem von Lebewesen ähneln. Es sind Modelle, die Eingangsdaten über Neuronen gewichtet kombinieren und daraus einen Ausgang errechnen. Neuronale Netze sind in der Technik zur Vereinfachung meist in mehreren Schichten aufgebaut.

Abbildung 2.1 zeigt ein einfaches Neuron mit einem Eingang. Ein Eingang wird mit dem Gewicht multipliziert um dann anhand der Übertragungsfunktion den Ausgang zu berechnen. Als Übertragungsfunktion dient meist die Sigmoid-Funktion, welche sich einfach differenzieren lässt und sich daher besonders gut eignet. Der Ausgang *out* dieses Neurons ergibt sich somit aus der dem Eingang *in* multipliziert mit dem Gewicht ω in der Sigmoid-Funktion

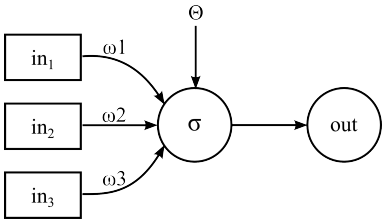


Abbildung 2.2: Neuron mit mehreren Eingängen

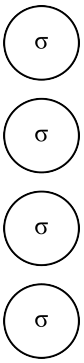


Abbildung 2.3: Layer von Neuronen

σ.

$$out = \sigma(\omega * in)$$

Ein Neuron hat in der Regel, wie in Abbildung 2.2 zu sehen, mehrere Eingänge. Außerdem wird noch ein Bias-Wert θ für die Sigmoid-Funktion hinzugefügt. Der Ausgang dieses Neurons kann somit wie folgt berechnet werden:

$$out = \sigma(\omega_1 * in_1 + \omega_2 * in_2 + \omega_3 * in_3 + \theta)$$

Um aus den einzelnen Neuronen ein Netzwerk zu machen, werden, wie in Abbildung 2.3 Layer gebildet. Ein Layer ist eine Menge an Neuronen, die untereinander nicht direkt verknüpft sind. Layer werden sequenziell hintereinander gehängt, wobei jedes Neuron eines Layers als Eingang für jedes Neuron des jeweils nächsten Layers dient. Ein solches Netzwerk ist auch in Abbildung 2.4 zu sehen. Dieses Netzwerk hat einen Layer für die Eingabedaten, einen für die Ausgabedaten und zwei innere Layer. Die inneren Layer werden als von außen Unsichtbar betrachtet und daher auch als Hidden-Layer bezeichnet.

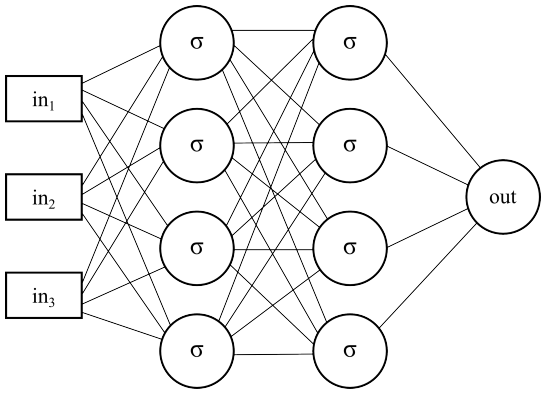


Abbildung 2.4: Ein Neuron

2.5 Überwacht / Unüberwacht

Die Algorithmen des Deep Learning werden im groben in zwei Kategorien geteilt, in die überwachten und die unüberwachten Methoden. Überwacht bedeutet, dass Eingangsdaten an das System angelegt werden, für die die gewünschten Ausgangsdaten bekannt sind. Das könnte zum Beispiel heißen, dass das Netz Bilder mit und ohne Gesichtern darauf bekommt und dann anhand der Richtigkeit des Ausgangs das Modell angepasst wird.

Diese Art des Lernens scheint logisch, benötigt aber sehr viele Kategorisierte Datensätze, die nicht immer verfügbar sind und tendiert leicht zur Überanpassung. Überanpassung bedeutet, dass das Netz die Trainingsdaten besser als notwendig lernt und für weitere Eingabedaten schlechtere Ergebnisse liefert als wäre es weniger trainiert worden.

Ein weitere Ansatz ist das unüberwachte Lernen, es handelt sich dabei um Algorithmen, die lediglich Eingangsdaten benötigen. Die Grundlegende Idee dabei ist es, das Netz Merkmale aus den Eingangsdaten erkennen zu lassen. Unüberwachtes Lernen ist besonders wegen der enormen Verfügbarkeit unkategorisierten Daten interessant und wird häufig als Grundlage vor dem überwachten Lernen eingesetzt. Es hilft einige Probleme des überwachten Lernens zu verbessern, so können die Gewichte aus dem unüberwachten Lernen als Startwert für ein überwachtes Lernen eingesetzt werden. Diese Gewichte haben mehr Aussagekraft über Merkmale der Eingangsdaten als zufällige Werte und können mit weniger kategorisierten Trainingsdaten oft bessere Ergebnisse erzielt werden.

Kapitel 3

Algorthmen

Fokus: viele Daten, parallele Verarbeitung, viel Rechenpower

3.1 Backpropagation

Backpropagation ist ein überwachter Algorithmus zum Trainieren von vereinfachten neuronalen Netzen bei denen sich der Ausgang auf einfachem Weg berechnen werde kann, so genannten Feedforward-Perzeptrons, zu sehen in Abbildung 3.1. Überwacht bedeutet, dass das Netzwerk beim lernen anhand von kategorisierten Eingabedaten bestimmte Ausgabedaten zu errechnen. Der Fehler der Berechnung des Ausgangs wird beim Backpropagation-Algorithmus in die Adaptierung der Gewichte rückgeführt. Der Trainingsprozess beginnt mit zufälligen Gewichten und arbeitet in den folgenden Schritten:

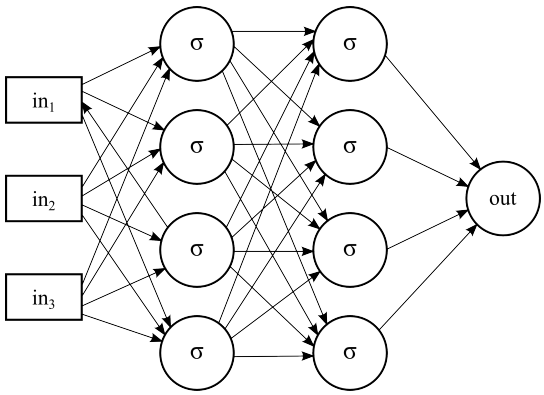


Abbildung 3.1: Ein Feedfowrward-Perzeptron

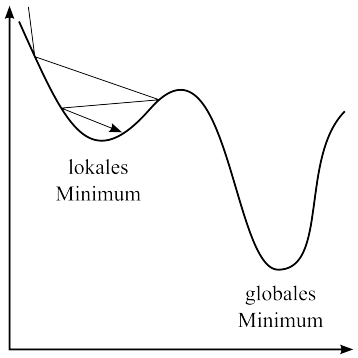


Abbildung 3.2: Lokales Minimum

1. Ausgang des Netzes für bestimmte Eingabedaten berechnen
2. Ausgang mit dem Sollwert vergleichen und daraus den Fehler errechnen

$$error = \frac{1}{2} \sum_{k \in K} (out_k - target_k)^2$$

$target$	...	Fehler, der später Einfluss in die Gewichte nimmt
out	...	Wert den der Ausgang erreicht hat
$target$	...	Wert den der Ausgang erreichen sollte
K	...	Anzahl der Neuronen in der Ausgangsschicht

3. Gewichte je nach Größe des Fehlers anpassen

Diese Schritte werden wiederholt, bis die Gewichte so angepasst wurden, das Eingabedaten möglichst gut klassifiziert werden können. Die Anpassung der Gewichte errechnet sich im wesentlichen anhand der Gradienten der Fehler und Gewichte. Zudem nimmt Schicht, vor der sich das jeweilige Gewicht befindet, Einfluss.

Wie der Name des Alogorithmus bereits verrät, wird der Fehler dabei im wesentlichen über eine mathematische Funktion in die Gewichte zurückgeführt. Diese Rückführung bringt das Problem von lokalen Minima mit sich. Wird die Änderung des Fehlers geringer, so wird auch die Anpassung geringer. Befindet man sich auf dem Weg in ein lokales Optimum, wie in Abbildung 3.2 zu sehen, so kann dieses sehr weit weg von dem globalen Optimum sein. Ist das Optimum entsprechend groß, so ist die Wahrscheinlichkeit, dass der Algorithmus in diesem Optimum hängen bleibt groß. Abhilfe können hier Verfahren wie der Bergsteigeralgorithmus oder zusätzliche Zufallswerte bei der Berechnung der Gewichte schaffen.

Der Algorithmus trainiert vordefinierte Ein- und Ausgabedaten mit dem Ziel für ähnliche Daten ähnliche Ergebnisse zu liefern. Trainiert man das Netz

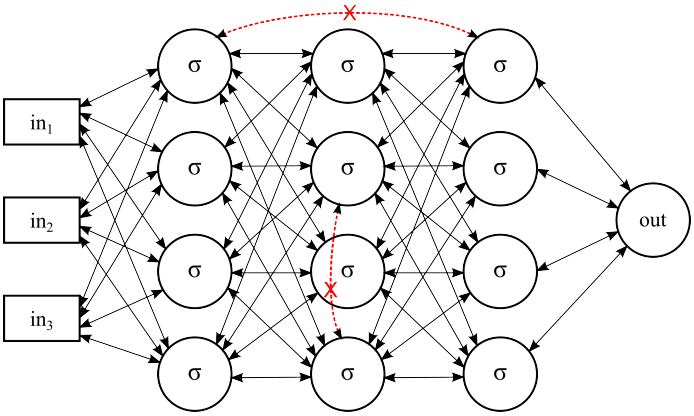


Abbildung 3.3: Eine Restricted-Boltzmann-Maschine

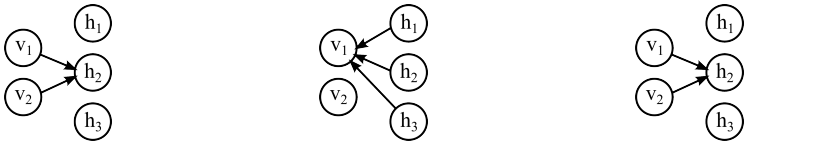
zu intensiv mit diesen Daten, so lernt es genau nur diese Trainingsdaten. Dies hat zur Folge, dass das Netz im späteren Betrieb schlechtere Ergebnisse liefert als hätte man es weniger lange trainiert.

3.2 Resctricted Boltzmann Maschines

Uneingeschränkte Neuronale-Netze sind schwierig und aufwendig zu trainieren. Restricted-Boltzmann-Maschines, im weiteren auch als RBMs bezeichnet, sind eingeschränkte neuronale Netzwerke. Wie in Abbildung 3.3 dargestellt, existieren hierbei nur Verbindungen zwischen aneinander liegenden Schichten. Das Modell erlaubt keine Verbindungen von Neuronen zu sich selbst und alle Verbindungen müssen gleichermaßen in beide Richtungen gehen. So lassen sich unter Festlegung der Werte einer Schicht direkt auf die nächste versteckte Schicht weiter gerechnet werden. RBMs arbeiten in der Regel mit bool'schen Werten, lassen sich durch Erweiterung aber auch mit anderen Zahlenräumen verwenden.

RMBs wurden bereits 1986 von Paul Smolensky unter dem Namen Harmonium erfunden, fanden jedoch erst 1998, rund 10 Jahre später, durch die Entwicklung von effizienten Lernalgorithmen durch Geoffrey Hinton Anwendung. ref

Zum trainieren von RMBs existieren verschiedene Algorithmen, die meist auf dem Prinzip beruhen, die Gewichte so anzupassen, das das hin und her Rechen zwischen zwei Schichten wieder die Ausgangsdaten ergibt. Im folgenden wird der Algorithmus von Geoffrey Hinton , der zugleich als der erste effizienten Lernalgorithmus für RMBs gesehen wird, erklärt. reference



(a) Versteckte Schicht berechen

(b) Von der versteckten Schicht zurück rechnen

(c) Erneut die versteckte Schicht berechnen

Abbildung 3.4: Drei Berechnungsschritte zum Lernen von RBMs.

3.2.1 Grundgedanke

Sind die Werte einer Schicht fixiert, so kann einfach auf die nächste versteckte Schicht weiter gerechnet. Zu beginn werden wird ein Trainingsdatenset an die Eingänge angelegt und die folgenden Operationen, wie auch in Abbildung 3.4 zu sehen, wiederholt ausgeführt:

1. Wahrscheinlichkeit p für die versteckten Neuronen h_j anhand der sichtbaren Neuronen v_i und der Gewichte w_{ij} berechnen
$$p(h_j = 1) = \frac{1}{1 + e^{-(b_j + \sum_i (v_i * w_{ij}))}}$$
2. Werte für die versteckte Schicht aus dem Mittelwert der Wahrscheinlichkeiten berechnen
3. Gradientenmatrix $< v_i h_j >$ über das dyadische Produkt errechnen
$$< v_i h_j >= v * y^T$$
4. Ausgehend von den errechneten versteckten Neuronen, zurück auf die sichtbare Schicht rechnen

Wiederholt man die in der Liste angeführten Schritte sehr oft, so pendeln sich für die Neuronen Werte ein, die im wesentlichen vom Modell und den den Wahrscheinlichkeiten abhängen, jedoch sehr wenig mit den Eingangsdaten zu tun haben. Anhand der Gradientenmatrix des letzten Durchlaufes lässt sich jedoch eine Distanz zum gewünschten Modell herausfinden und so können die Gewichte wie folgt angepasst werden:

$$\Delta\omega_{ij} = \varepsilon(< v_i h_j >^0 - < v_i h_j >^\infty)$$

Diese Art der Berechnung lieferte sehr gute Modelle. Durch die vielen Iterationen benötigt der Algorithmus sehr viel Rechenzeit und ist daher nicht praxistauglich. Zudem ist es schwierig festzustellen, wie viele Iterationen bis zum Einpendeln notwendig sind.

3.2.2 Abkürzung

Der oben genannte Algorithmus lässt sich in seiner Komplexität erheblich reduzieren, in dem die versteckte Schicht lediglich zwei mal berechnet wird. Geoffrey Hinton hat mit der Vereinfachung gezeigt, dass es möglich ist, bereits mit dem Vergleich der ersten und zweiten Gradientenmatrix möglich ist gute Ergebnisse zu erzielen. Diese Methode wird *Contrastive Divergence* genannt.

Die Regel zur Anpassung der Gewichte lautet dann:

$$\Delta\omega_{ij} = \varepsilon(< v_i h_j >^0 - < v_i h_j >^1)$$

3.2.3 Begründung

Der Grundgedanke von *Contrastive Divergence* ist, dass das Modell mit Zufallsgewichten weg von den Eingabedaten, hin zu Daten die ihm besser gefallen, wandert. Wenn man erkennt wo hin das Modell die Daten ändert, kann man die Gewichte so adaptieren, dass dem Modell die Eingabedaten am besten gefallen.

3.2.4 Eignung

Das Modell besitzt nicht genügend Neuronen um alle Eingabedaten zu speichert und muss daher, um die Eingabedaten tatsächlich reproduzieren zu können, etwas aus den Eingabedaten lernen. Es eignet sich daher, um aussagekräftige Merkmale aus unkategorisierten Daten zu extrahieren.

3.3 Deep Autoencoders

Deep Autoencoder sind neuronale Netze aus mehreren Schichten, die sich in zwei wesentliche Teile trennen lassen. Im ersten Teil, dem Encoder, sinkt die Anzahl der Neuronen mit jeder Schicht, während sie im zweiten Teil, dem Decoder wieder Steigt, bis am Ende wieder die Anzahl der Eingangsparameter erreicht ist. Abbildung 3.5 zeigt ein solches Netz. Im inneren hat das Netz weniger Neuronen und somit Speichermöglichkeit als Eingangsparameter. Es wird versucht das Netz so zu trainieren, dass es am Ausgang die gleichen Daten berechnet wie am Eingang angelegt entstehen. Gelingt das Training, so muss das Netz Merkmale etwas aus den Daten gelernt haben.

Richtig trainiert, komprimiert das Netz die Daten also. So wäre es zum Beispiel möglich Bilder durch das Netz zu schicken, die kleinste Schicht zu speichern und jederzeit über den Decoder wiederherzustellen. Für Bilder, Ton und Videos gibt es einige Kompressionsalgorithmen die sehr fein abgestimmt

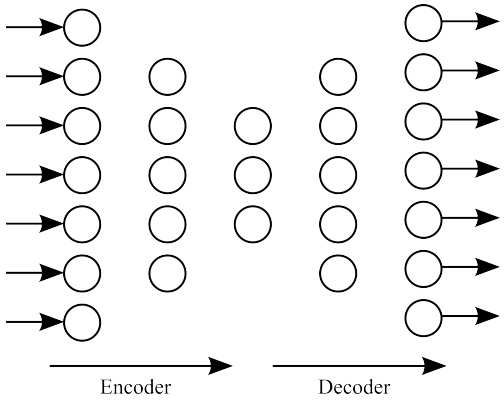


Abbildung 3.5: Deep Autoencoder

die Charakteristik menschlicher Organe berücksichtigt und daher in der Regel wesentlich bessere Ergebnisse liefern. Dennoch demonstriert das Beispiel, dass damit unter Umständen erheblich Speicher und Verarbeitungsleistung gespaart werden kann.

Deep Autoencoder liefern sehr vergleichbare Ergebnisse und werden daher gerne als Benchmark für Deep Learning-Algorithmen verwendet. Deep Autoencoders tendieren zu Unteranpassung (eng. underfitting), da nicht genügend Neuronen verfügbar sind um alle Möglichkeiten abzubilden. Eine einfache und effektive Variante ein solches Netz zu trainieren ist es, jeweils zwei Layer separat als restricted Boltzmann Maschine anzusehen.

Lernt man solche Algorithmen mit mit Bildern und visualisiert die entstandenen Gewichte, so erkennt man, dass solche Algorithmen in erster Instanz meist Kanten und Farbintensitäten in Bildern finden. In der zweiten Schicht findet man einfache Kombinationen dieser Elemente und ab der dritten Schicht werden meist schon sehr brauchbare Neuronen zur vollständigen Objekterkennung ausgebildet.

3.4 Faltungscodierte neurale Netze

Faltungskodierte neuronale Netze (eng. Convolutional Neural Networks) sind neuronale Netze die zur Reduktion der Gewichte in einzelne Teile zerteilt sind. Bei den bisher angeführten Netzen wurden jeweils alle Neuronen einer Schicht mit jedem Neuron der nächsten Schicht verbunden. Der Gedanke bei faltungskodierten Netzten geht man davon aus, dass weit voneinander entfernte Neuronen kaum etwas miteinander zu tun haben, während nahe aneinander liegende Neuronen Zusammenhänge besitzen. Es werden daher Gruppen über aneinander liegende Neuronen gebildet, die untereinander in

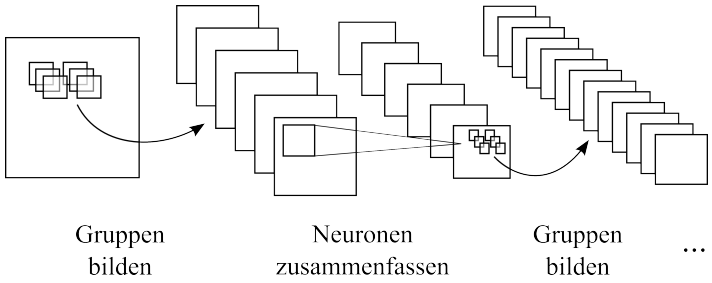


Abbildung 3.6: Faltungscodiertes neuronales Netz

die nächste Schicht verknüpft sind. Wie am Beispiel eines Bildes gut erkennbar, ist es gut möglich, dass ein Merkmal nicht genau in eine dieser Gruppen passt. Es werden daher viele überlappende Gruppen gebildet.

Im nächsten Schritt gibt es eine zusammenfassende Funktion, die die Ergebnisse aneinander liegender Werte für die nächste Schicht mit einer mathematischen Funktion zusammenfasst. Diese Funktion könnte in einer Schicht zum Beispiel die Rotation eines Bildes ignorieren in dem es Kanten in mehrere Richtungen zusammenfasst. Ein faltungskodiertes neuronales Netz wiederholt genau diesen zwei Komponenten, die überlappende Gruppierung von Neuronen so wie das zusammenfassen von Neuronen auf jeder Schicht. Abbildung 3.6 zeigt ein solches Netz.

Dieser Typ von Netzwerk ist bereits 1980 bekannt, wurde in den folgenden Jahrzehnten optimiert und erlebte einen Aufschwung durch parallelisierte Berechnung auf einer GPU . Nach weiteren Optimierungen in den vergangen Jahren wurde ein solches Netzwerk unter anderem für ein Projekt von Google eingesetzt, das nicht zuletzt das so genannte face-neuron ausprägte.

paper Kunihiko fukushi-ma

Dan Ci-resan

google paper finden

3.5 Sparse coding

Sparse coding ist ein Algorithmus zum trainieren eines neuronalen Netzes mit dem Ansatz möglichst aussagekräftige Merkmale zu finden. Anders als beim Deep Autoencoder wird hier nicht nur berücksichtigt wie gut das Ergebnis wiederhergestellt werden konnte, sondern auch wie eindeutig das Ergebnis war. Ziel ist es an möglichst vielen stellen im Ausgangsvektor 0er zu haben und dennoch sehr gute Ergebnisse zu finden. Der Fehler setzt sich aus diesen beiden Faktoren zusammen und wie folgt berechnet:

$$min\frac{1}{T}\sum_{t=1}^{T} min\frac{1}{2}|x^{(t)} - D * h^{(t)}| + \lambda|h_1^{(t)}|$$

$x^{(t)}$	...	Eingangswert
$h^{(t)}$	...	Ausgangswert
D	...	Anzahl der Ausgänge
λ	...	Steuert wie stark sich die 0er im Ausgangsvektor auswirken

Die zwei Bedingungen widersprechen sich teilweise und können über λ balanciert werden. Die Berechnung der Gewichte für die versteckten Schichten ist beim Sparse Coding wesentlich aufwendiger als bei Deep Autoencodern, da es sich hier um ein Optimierungsproblem und nicht nur um eine lineare Funktion handelt.

Kapitel 4

Anwendungen

In den vergangen Jahren haben sich Unternehmen zunehmend an vielschichtigen neuronalen Netzten interessiert. Firmen wie Google, Appleund Microsoft stellten renommierte Wissenschaftler ein und kaufen Unternehmen in diesem Bereich.

einstellung ref

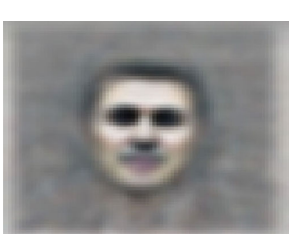
link

4.0.1 Google Projekt

In einem Projekt2012 trainierte Google ein Netz mit 10 Millionen Bilder aus dem Internet. Ein paar Eckdaten dazu:

http://research.google.

- 10 Millionen Trainingsbilder mit je 200x200 Pixel
- Autoencoder mit sparse coding
- 9 versteckte Schichten



(a) menschliches Gesicht

(b) Gesicht einer Katze

(c) menschlicher Körper

Abbildung 4.1: Markante Ausprägungen durch das lernen von Bildern aus dem Internet

- 1 Milliarde Verbindungen
- Training mit asynchronem stochastischem Gradientenvefahren
- Auf auf 1000 Maschinen mit jeweils 16 Rechenkernen drei Tage lang trainiert

Bekannt wurde das Projekt vor allem durch die Ausprägung des in Abbildung ?? zu sehenden Gesichts- und Katzen-Neurons. Es wurde mit relativ kleinen Bildern, dafür aber mit sehr vielen unbeaufsichtigt trainiert. Ein späterer Benchmark auf ImageNET, eine Datenbank mit Kategorisierten Bildern, ergab 15.8% Trefferquote, das entspricht einer Steigerung des Spitzenwertes um 70%.

Aus technischer Perspektive ist, die das Netz und damit die verwendete Technik weniger herausragend als die Menge der Trainingsdaten und die damit verbunden Rechenleistung.

4.1 Google Street View

Google trainierte durch ihre Projekt Street View ein tiefes faltungskodiertes Netz das lernte Hausnummern zu erkennen. Für die Adresssuche auf Google Maps und dem damit verbundenen Navigationssystem ist es von wesentlicher Bedeutung Hausnummern richtig identifizieren zu können, besonders in Gebieten, bei denen Hausnummern nicht fortlaufend sind.

Google trainierte mehrere verschiedene Netze mit unterschiedlicher Anzahl an versteckten Schichten. Es stellte sich heraus, dass das tiefste Netz, mit elf versteckten Schichten die Hausnummern am besten erkennen konnte. Das Netz wurde mit über 100 Millionen hochauflösenden Bildern trainiert und forderte entsprechend viel Rechenleistung. Zum Training wurde daher der DistBeliefAlogrithmus verwendet, ein Alogorithmus der sich besonders gut verteilt rechnen lässt.

Nummern die das Netz nicht auflößen konnte, haben unbewusst Menschen gelöst. Hierzu hat Google ein Projekt ins Leben gerufen, das Besucher von Webseiten an sicherheitsrelevanten Stellen einen Code identifizieren lässt, um sicherzustellen dass es sich dabei tatsächlich um einen Menschen und keinen Bot handelt.

Insgesamt hat das Netz bemerkenswert gut abgeschnittenund erkennt mittlerweile 97,84% aller Hausnummern wie sie in Abbildung 4.2 zu sehen sind und sogar 99,8% aller Sicherheitscodes der schwierigsten eingesetzten Art, der *reCAPTCHA*. Das Netz ist somit besser als die meisten Menschen, wo-

bild quelle: http://www.extremete
google-has-built-a-neural-network-to-identify-100-million-house-numbers-for-streetview

link zu DistBelief algorithmus

http://arxiv.org/abs/1

Abbildung 4.2: Hausnummer die das neuronale Netz aus Google Street View erlernte.

durch fraglich ist wie gut diese Sicherheitscodes in Zukunft tatsächlich vor Bots schützen werden.

4.2 Spracherkennung

Auch bei der Spracherkennung werden neuronale Netze in Kombination mit Deep Learning immer stärker und lösen nach und nach die Methode der Gaussian Mixture Models (GMM-Methoden) ab. Neuronale Netze für die Spracherkennung von Tonspuren funktionieren sehr ähnlich zu denen für die Bilderkennung. Wie bei der Bilderkennung prägen sich auf den ersten Schichten einfache Merkmale wie Konsonanten und Vokale aus. In höheren Schichten entstehen dann aussagekräftigere Merkmale bis hin zu vollständigen Wörtern, Phrasen und Stimmungen.

Ein heute bereits aktives Feld der Anwendung ist die Sprachsteuerung von Geräten, die quasi auf allen großen Plattformen verfügbar ist. Durch die große Menge an verfügbaren Sprachdaten von Mobiltelefonen, die zudem sehr gut kategorisierbar sind, ist zu erwarten, dass diese Netze in Zukunft noch stark an Bedeutung gewinnen werden.

links siri, coratana, google now

4.3 Hardware

2012 Hat Google einen Rekord für das größte neuronale Netz, mit 1.000 Servern und 16.000 Rechenkernen aufgestellt. Bereits 2013 Hat NVIDIA gemeinsam mit der Stanford Universität ein 6,5-fach größeres neuronales Netz auf GPUs realisiert. Das Netz arbeitet auf nur 16 Computern mit jeweils vier Hochleistungs-Grafikkarten aus dem Konsumentenbereich und ist somit wesentlich günsti-

referenz

ger.

Momentan entwickelt Qualcomm an Controller der eine neuronale Recheneinheit enthält. Der wesentliche Unterschied zu einer herkömmlichen CPU liegt dabei darin, dass alle Neuronen einer Schicht parallel rechnen. Das Netz benötigt daher nur wenige Zyklen bis zum Ergebnis, während herkömmliche CPUs jedes Neuron hintereinander berechnen muss. In den frühen Entwicklungsstufen konnte das Netz bereits einige Bilderkennungsaufgaben gleich gut wie komplexe mathematische Algorithmen lösen. Der Chip soll 2014 in die Massenproduktion gehen und unter anderem in Mobiltelefonen zur Verfügung stehen.

4.4 Weitere Anwendungen

Verkehrsschilder erkennen	Traffic sign recognition (99,46prozenz, besser als menschen! gewinner h

Kapitel 5

Zusammenfassung

In der Arbeit wurden die wesentlichen Konzepte und Algorithmen rund um das Deep Learning aufgearbeitet. Deep Learning-Algorithmen basieren im Wesentlichen auf sehr ähnlichen Prinzipien und wurden mit der Entwicklung der Technik an diese angepasst. So sind moderne Algorithmen, wie der zum Beispiel der DistBelief-Algorithmus, im wesentlichen darauf ausgelegt eine parallelisierte Berechnung zu ermöglichen und somit die Hardware bestmöglich auszunutzen.

Aktuelle Netze sind meist sehr vielschichtig und werden mit unbeaufsichtigtem Lernen vorab trainiert. Aufwendiges Feature-Engineering durch Expertenwissen kann meist schon entfallen und wird automatisch durch das unbeaufsichtigte Lernen gemacht.

Bereits heute werden aktiv Netze zur Erkennung von Verkehrsschildern, Hausnummern und Gesichtern eingesetzt. Im Grundsatz zeigen momentan Benchmarks dass die Ergebnisse von neuronalen Netzen im wesentlichen von der Menge der Trainingsdaten und der Größe des Netzes abhängen. Mit zunehmender Vernetzung und dem Internet existieren in vielen Bereichen Daten in sehr großen Mengen. Mit dem Fortschritt der Technik und zunehmendem Interesse an diesem Forschungsgebiet, sind in naher Zukunft noch viele Interessante Anwendungen zu erwarten.

Persönlich finde ich die Entwicklungen in Bereich der Chipherstellung besonders interessant. Die Geschichte hat bereits mehrmals gezeigt, dass die Berechnung auf CPUs über Grafikkarten bis hin zu dedizierten Chips enorme Leistungssprünge bringt. So dürfte auch die Entwicklung von Chips wie sie zum Beispiel Qualcomm momentan entwickelt einige Neuerungen bringen.

Die Leistungsfähigkeit neuronaler Netze ist schwierig zu verstehen und so entsteht immer wieder die etwas philosophische Frage, in wie weit ein solches Netz an die Leistungsfähigkeit des Menschen kommt oder ihn gar eines Tages Konkurrenz macht. Gut Trainiert schlagen diese Netze Menschen schon heute

in der ein oder anderen Aufgabe. An dieser Stelle ist zu erwähnen, dass bereits um 1940 die *Zuse* Maschinen von *Konrad Zuse*, die ersten Rechner der Welt, den Menschen in vielen Rechenaufgaben geschlagen haben. Bis neuronale Netze so groß werden, dass sie sich ein Bild von der Natur machen können, dürfte noch eine ganze Zeit vergehen.

Literaturverzeichnis

1 **Blank u. a. 2008** Blank, Michael ; Bopp, Thomas ; Hampel, Thorsten ; Schulte, Jonas: Social Tagging = Soziale Suche? In: Gaiser, Birgit (Hrsg.) ; Hampel, Thorsten (Hrsg.) ; Panke, Stefanie (Hrsg.): *Good Tags - Bad Tags: Social Tagging in der Wissensorganisation*, Waxmann, 2008, S. 85–96