Proseminar: „Softwareentwicklung in der Wissenschaft“

Betreuer: Tobias Finn

SoSe 2020

GANs in der Wissenschaft

Johannes Kolhoff

Name: Johannes Kolhoff

Matrikel Nummer: 7260796

Studiengang: B. Sc. Informatik

Fachsemester 2

Abgabedatum: 31.08.2020

Inhalt

1 Einleitung

Die folgende Arbeit beschäftigt sich mit Generative Adversarial Networks (GANs) und deren Anwendbarkeit in der Wissenschaft. GANs werden dabei als Konzept theoretisch erklärt und ich werde nicht auf spezifische Implementationen eingehen. Durch zwei Beispiele aus der Wissenschaft werden die Stärken und Schwächen deutlich und ich werde eine Bewertung von GANs als wissenschaftlicher Methode geben.

2 Funktionsweise von GANs

GANs sind eine neue Methodik und wurden erstmals von Ian Goodfellar in einem Paper aus dem Jahr 2014 vorgestellt [1]. GANs steht für Generative Adversarial Networks, also Erschaffene Gegenspieler Netzwerke und sind eine Art von unüberwachtem Lernen.

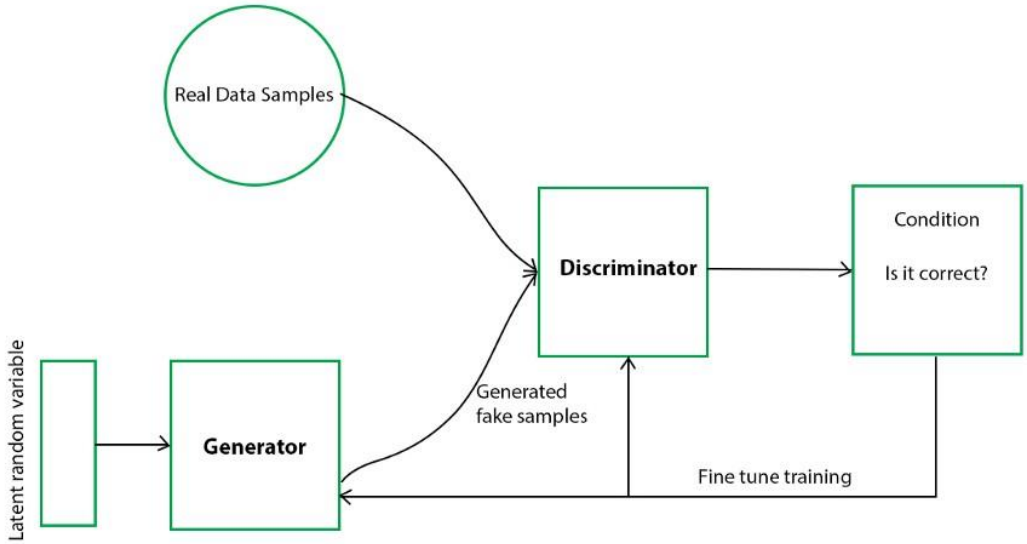

Abbildung 1 Schematische Darstellung der Funktion eines Generative Adversarial Networks. Quelle: Pal, Rajtilak (2020) „Deep Learning: The Fastest Growing Tech. in the World“ Medium https://mc.ai/generative-adversarial-network/ [24.8.2020]

Die obige Abbildung stellt die generelle Funktionsweise eines Generative Adversarial Networks da. Die Komponenten der Abbildung werden im Folgenden einzeln erklärt.

2.1 Generator

Der Generator ist ein neuronales Netzwerk, welches den Input beziehungsweise den Noise nutzt, um einen neuen Datensatz zu generieren. Durch das Feedback, das der Generator vom Discriminator bekommt, werden die generierten Daten besser. Der Generator selbst kennt also die echten Daten aus dem Datensatz der Referenzdaten nicht, sondern nähert sich diesen durch Ausprobieren und Feedback an. [1]

2.2 Discriminator

Der Discriminator bekommt ein Datum des Datensatzes, also eine echte Referenz, und ein vom Generator erzeugtes Datum. Der Discriminator soll entscheiden, welches das authentische Datum ist und welches vom Generator erzeugt wurde. Der Discriminator verteilt eine Wahrscheinlichkeit von eins auf beide Daten. Verteilt er eine Wahrscheinlichkeit von 0 für das eine Datum und 1 für das andere Datum, würde es bedeuten, dass der Discriminator sich sehr sicher ist, welches das Original ist. Bei einer Verteilung von 0,5 und 0,5 würde es bedeuten, dass der Discriminator es bei beiden für gleich wahrscheinlich hält, dass sie das Original sind. [1]

2.3 Condition

Zwei Netzwerke namens Generator und Discriminator „spielen“ gegeneinander und optimieren sich dadurch gegenseitig. Dem Generator wird ein Input z gegeben, aus dem er ein Output g(z) generiert. Der Discriminator bekommt das Output g(z) des Generators und eine Einheit des Datensatzes. Der Discriminator entscheidet, welches der beiden Daten das Original ist und welches Datum vom Generator erschaffen wurde. Er berechnet eine Funktion d(g(z)) für das vom Generator erzeugte Datum. Siehe hierzu Abbildung 2. (Außerdem rechnet er eine Funktion für das Originaldatum aus dem Datensatz, beide Wahrscheinlichkeiten zusammen ergeben 1. Daher werde ich die berechnete Funktion für das Original nicht weiter erwähnen, da sie für die Bewertung des Generators irrelevant ist.) Anschließend bekommen der Discriminator und Generator ein Feedback in Form der Fehlerfunktion. Der Discriminator bekommt dabei das binäre Feedback in Form der Auflösung, welches der beiden Daten von den Originaldaten stammt und welches vom Generator erzeugt wurde. Währenddessen erhält der Generator das differenzierte Feedback, beziehungsweise die Bewertung des Discriminators. Der Generator will eine möglichst hohe Wahrscheinlichkeit, also von 1 als Bewertung bekommen, während der Discriminator den

Generator (richtigerweise) möglichst niedrig einschätzen will. Die Discriminator-Funktion d(g(z)) für das generierte Datum wird vom Generator maximiert und vom Discriminator minimiert. So ist der Antagonismus gewährleistet. Nach ausreichender Trainingszeit sollte der Generator so gut sein, dass der Discriminator Original und Fälschung nicht mehr unterscheiden kann. Er berechnet deswegen eine Wahrscheinlichkeit von 0,5 für Original und Fälschung. [2]

2.4 Kritik an der Abbildung

Wie in 2.3 beschrieben, nutzt der Generator als Feedback das differenzierte Feedback, welches er direkt vom Discriminator erhält, während der Discriminator ein binäres Feedback in Form der Auflösung bekommt. Dem Discriminator wird also mitgeteilt, was das Original und was das generierte Datum ist. Er kann dieses mit seiner Vorhersage verrechnen, indem er die Abweichung ausrechnet und so ein differenziertes Feedback generiert. Für diese Fehlerfunktion ist keine aufwendige Berechnung und kein eigenes neurales Netz notwendig. Dies ist in der Abbildung 1 missverständlich.

Bessere Abbildung der Fehlerfunktion:

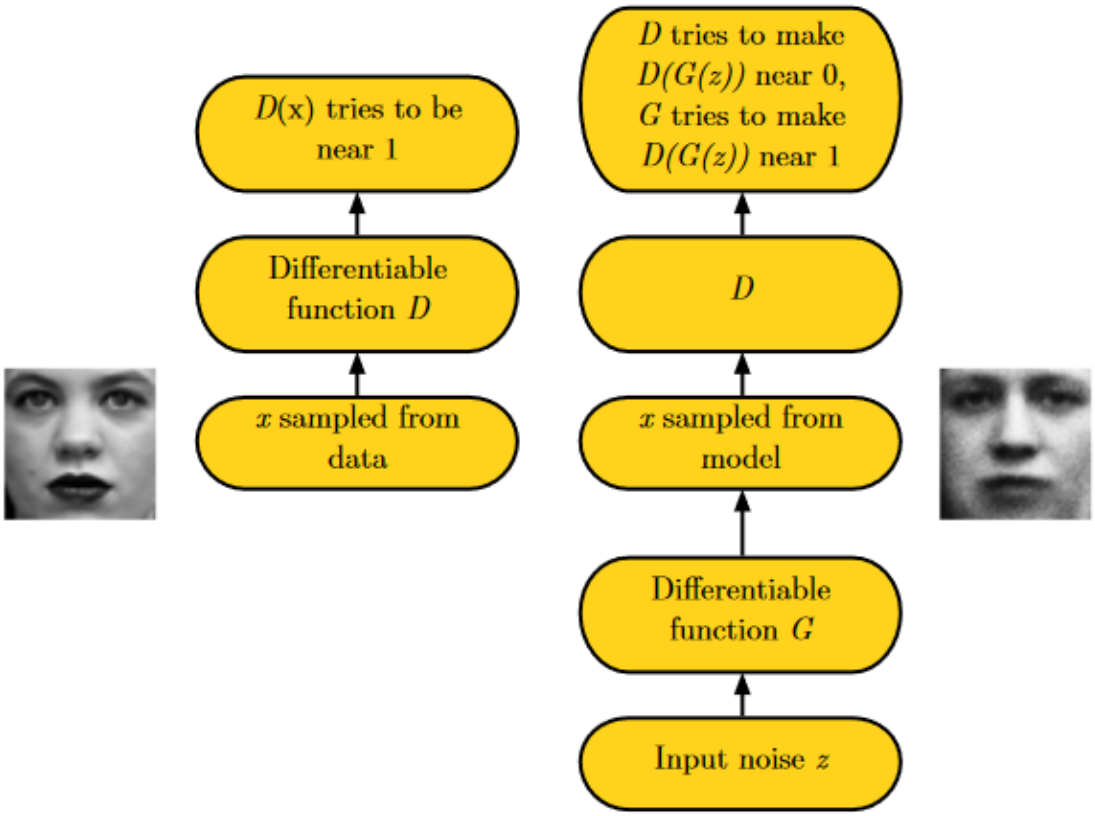

Abbildung 2: Beschreibung des Prozesses des Lernens mit Fehlerfunktion als Flussdiagramm. Es ist so zu lesen, dass der linke und rechte Teil auf einer Ebene gleichzeitig ausgeführt wird und der rechte Teil früher anfängt. [2]

Die Abbildung 2 beschreibt die Fehlerfunktion mathematisch richtiger und zeigt, welche Ziele die beiden Netzwerke verfolgen.

2.5 Datensatz

Der Datensatz (in Abbildung 1 als Real data sampels bezeichnet) wird vor der Trainingsphase zusammengestellt. Die Komposition und die Qualität der Daten, die zum Training benutzt werden, entscheiden maßgeblich über die Qualität der Ergebnisse. (garbage in garbage out) [5]

Die Qualität der Daten in Bezug auf Größe, z.B. Auflösung von Bildern, bestimmt über die Schwierigkeit der Aufgabe und somit die Dauer der Trainingsphase. Wenn beispielsweise sehr hochaufgelöste Fotos genutzt werden, dauert es länger, bis der Generator ähnliches erschaffen kann als bei reduzierten Darstellungen. Bei begrenzter Rechenkapazität kann beispielsweise bei Bildern die Auflösung begrenzt werden, eine Angleichung der Größe z.B. Auflösung aller Elemente des Trainingssets ist auf jeden Fall sinnvoll.

Die Qualität der Daten in Bezug auf die Auswahl der Daten bestimmt über die Qualität des Ergebnisses. Bei einer gleichförmigen Auswahl an Daten werden im Allgemeinen bessere Ergebnisse erzielt als bei einer sehr diversen Auswahl. Wenn es Spezialfälle gibt, ist eine Möglichkeit, sie wegzulassen, eine andere wäre, genug Beispiele des Spezialfalls hinzuzufügen, sodass ein Muster erkannt werden kann. Allgemein ist eine Limitierung auf Teilbereiche sinnvoll, z.B. Gesicht statt ganzer Körper bei Fotos von Menschen.

2.6 Input

Dem Generator wird eine Eingabe (in Abbildung 1 als latent random variabels bezeichnet) gegeben, die er nutzt, um aus ihr ein Datum zu generieren. Diese Eingaben sind insbesondere bei der Bildverarbeitungs-Anwendung oft Random Noise; zufällige Variablen oder Vektoren können aber auch genutzt werden. Bei wissenschaftlichen Anwendungen können unvollständige Datensätze genutzt werden.

GANs werden meist dazu genutzt, um zu einem vorgegebenen Datensatz ähnliche, aber neue Daten zu generieren. Dabei würde man sich also nach abgeschlossener Trainings-Phase den Generator zunutze machen. Aber auch eine Nutzung des Discriminators ist denkbar, um authentische Daten erkennen zu können. Beispielsweise könnte ein trainierter Discriminator bestimmte Objekte, wie z.B. Tiere auf Bildern erkennen.

3 Erstes Beispiel aus der Wissenschaft: GANs angewendet auf die Bildverarbeitung von Galaxienbildern

3.1 Problemstellung/Ausgangslage

Für die Verbesserung von Fotografien von Galaxien sollte eine Bildbearbeitungs-Software entwickelt werden. Dabei sollte die Funktion über eine herkömmliche Bildverbesserungs-Software hinausgehen, in dem Sinne, dass die Galaxie und andere kosmische Objekte als solche erkannt und berücksichtigt werden. Zum Erschaffen dieses Programmes wurde kein neues Programm geschrieben, sondern ein vorhandenes trainiert und der trainierte Generator sollte als dieses Bildverbesserungs-Programm genutzt werden können. In diesem

Fall war das Ziel nicht, mit der GANs vollkommen neue Bilder zu generieren, sondern den Generator auszukoppeln und als eigenständiges Programm unabhängig zu nutzen. [5]

3.2 Trainingsphase

Das Setup dieses Generative Adversarial Networks unterscheidet sich von einem Standardsetup, in dem der Discriminator in diesem Fall drei Eingaben bekommt: ein hochauflösendes Originalfoto einer Galaxie, eine verpixelte beziehungsweise herunterskalierte Version des gleichen Fotos und das vom Generator aus dem herunterskalierten Foto erzeugte Bild. Noch vor der Trainingsphase werden also die Daten vorbereitet, indem man die Originalbilder herunterskaliert. Dem Generator wird dann jeweils ein herunterskaliertes Bild gegeben, aus dem er ein neues generieren soll. Der Discriminator bekommt dann das Originalbild, die herunterskalierte Version desselben Originalbildes und das von dem Generator aus dieser herunterskalierten Version der gleichen Originalbildes erzeugte neue Bild. Demnach soll, wenn die Trainingsphase abgeschlossen ist, der Generator aus dem herunterskalierten Bild ein neues Bild erzeugen, das nicht mehr vom Original zu unterscheiden ist. [5]

3.3 Bewertung

Die Wiederherstellung von diesen Galaxienbildern funktioniert mit dem Generator des GANs deutlich besser als es bei konventionellen Methoden der Fall sein würde. Es gibt aber dennoch einige Probleme. So werden sehr seltene Objekte nicht erkannt und dementsprechend auch nicht richtig rekonstruiert. Des Weiteren wird zu jeder Eingabe eine realistisch aussehende Galaxie generiert, auch wenn auf der Eingabe keine Galaxie zu sehen ist. Es gibt keine Möglichkeit, eine Eingabe abzulehnen, wenn sie von zu schlechter Qualität ist oder das Foto etwas ganz anderes zeigt. Durch diese Nachteile sind die generierten Bilder mit Vorsicht zu betrachten, da man nie genau weiß, ob etwas rekonstruiert oder neu hinzugefügt wurde. Man sollte sie nicht als absolut und korrekt ansehen. Ein weiteres Problem physikalischer Natur ist, dass die GANs nur mit Galaxien trainieren können, von denen es bereits hochauflösende Bilder gibt. Da weit entfernte Galaxien durch die Rotverschiebung anders aussehen, bräuchte man von ihnen zunächst hochauflösende Fotos, mit denen man die GANs trainieren könnte, bevor man Fotos von solchen Galaxien verbessern kann. Diese Methode erzeugt also keine wirkliche Erweiterung des Horizonts in

dem Sinne, dass man jetzt weiter ins All blicken kann, aber sie ist eine Möglichkeit, Bilder von beispielsweise nicht so guten Teleskopen zu verbessern. [5]

4 Zweites Beispiel aus der Wissenschaft: GANs angewendet bei der Verarbeitung von Messdaten am LHC im CERN

4.1 Problemstellung/Ausgangslage

Das zweite Beispiel befasst sich mit dem Teilchenbeschleuniger Large Hadron Collider (LHC) an der Europäischen Organisation für Kernforschung (CERN). Nach einer Kollision hinterlassen Teilchen Spuren, die von Detektoren erfasst werden. Aus diesen Spuren lassen sich die Flugbahnen durch mathematische Berechnungen rekonstruieren. In den nächsten Jahren wird sich durch Verbesserungen der Sensoren die Menge der gesammelten Daten verdreißigfachen. [3] Diese Menge an Daten ließe sich nicht mit der konventionellen Methode verarbeiten, da der Output an Messdaten die Rechenkapazität überstiege. Da diese herkömmlichen Berechnungen sehr aufwendig sind und sehr viel Rechenzeit benötigen, wurde nach einer Alternative gesucht.

4.2 Trainingsphase

Es wurde ein Generative Adversarial Networks eingesetzt, um - wie im ersten Beispiel - den Generator nach Abschluss der Trainingsphase als eigenständige Software nutzen zu können. Als Input des Generators wurden alte Messdaten genutzt, zu denen es schon mit der konventionellen Methode berechnete Flugbahnen gab. Dem Generator wurden dann die rohen Messdaten gegeben und der Discriminator bekam die dazu passenden berechneten Daten. [3]

4.3 Ergebnisse

Im Ergebnis konnte eine gute Approximation erreicht werden. Dadurch lassen sich die Messdaten schnell in die wahrscheinlichen Daten der Flugbahn umrechnen. So können die Berechnungen in einem Bruchteil der Zeit, die die konventionelle Methode benötigt, gemacht werden. Dadurch können alle Daten in kürzerer Zeit verarbeitet werden. Der große Nachteil ist, dass die Ergebnisse nicht mathematisch bewiesen sind, sondern lediglich eine Approximation sind. Eine Anwendung des konventionellen Algorithmus ist also immer noch

sinnvoll, aber in Anbetracht der Tatsache, dass man mehr Messdaten hat, als man sie verarbeiten kann, kann es trotzdem sinnvoll sein, diese Methode zu nutzen, um die Messdaten einer ersten Analyse zu unterziehen. [3]

5 Diskussion/Bewertung

Ein großer Vorteil von GANs ist der relativ geringer Aufwand bei der Nutzung. Da es sich bei GANs um unüberwachtes Lernen handelt, benötigen sie nach dem Setup nur noch Rechenzeit und keine menschliche Unterstützung mehr. Dies steht im starken Kontrast zu herkömmlichen Softwareprojekten. So kann ein Code erschaffen werden, ohne dass er manuell programmiert werden muss. Dies birgt aber auch die Gefahr, dass er nicht so funktioniert, wie man es sich vorgestellt hat, und nicht alle Anwendungen sind am Ende erfolgreich. Dies gilt aber auch für die Softwareentwicklung im Allgemeinen. Dadurch, dass der (menschliche) Aufwand bei dem Training von GANs gering ist, ist das Risiko geringer als bei konventioneller Softwareentwicklung.

Das große Problem von GANs in der Wissenschaft ist, dass die klassische Anwendung des Generierens in der Wissenschaft oft nicht sinnvoll ist. Messdaten und Ergebnisse aus Experimenten müssen exakt sein und generierte Messdaten ergeben in den seltensten Fällen Sinn. Da man den Prozess des Generierens oft nicht nachvollziehen bzw. beweisen kann, müssten alle Ausgaben eines Generators überprüft werden. Bei herkömmlicher Anwendung von GANs auf Fotos ist das relativ einfach, da man schnell sehen kann, ob es „gut“ aussieht oder nicht. Bei abstrakten Messdaten ist das meistens nicht möglich. Ein weiteres Problem ist, dass oft aus verschiedenen vom Generator erschaffenen Daten ‚Cherry picking‘ betrieben wird, also besonders gute Ergebnisse präsentiert werden. Dadurch werden die Schwächen des Generators versteckt. Solch ein ‚Cherry picking‘ ist in der Wissenschaft generell keine gute Praxis. GANs bergen also auch die Gefahr, eine Korrektheit zu suggerieren, die nicht existiert.

Anzumerken ist hier noch der Aspekt der Inputvektoren, die mit Attributen des Datensatzes korrespondieren. Auf diesen Aspekt wurde in dieser Arbeit nicht eingegangen, obwohl diese Beziehung ist auch für wissenschaftliche Anwendungen relevant ist. Außerdem ist anzumerken, dass GANs immer noch eine vergleichsweise neue Methode sind und sich in den nächsten Jahren weiterentwickeln werden und sich so neue Anwendungsbereiche ergeben, die auch relevanter für die Wissenschaft sein können.

6 Literaturverzeichnis:

1. Goodfellow, Ian J / Pouget-Abadie, Jean
   (2014): “Generative Adversarial Networks“
   https://arxiv.org/abs/1406.2661

2. Goodfellow, Ian J
   (2016): “NIPS 2016 Tutorial: Generative Adversarial Networks”
   OpenAI, ian@openai.com
   https://arxiv.org/pdf/1701.00160.pdf

3. Musella, Pasquale / Pandolfi Francesco
   (2018) “Fast and accurate simulation of particle detectors using Generative adversarial networks”
   https://arxiv.org/pdf/1805.00850v2.pdf

4. Radford, Alec / Lucke Metz
   (2016) “Unsupervised reprentation learning with deep convolutional Generative Adversarial Networks”
   https://arxiv.org/pdf/1511.06434.pdf

5. Schawanski, Kevin / Zhang, Ce
   (2017) “Generative Adversarial Networks recover features in astrophysical images of galaxies beyond the deconvolution limit”
   https://arxiv.org/abs/1702.00403