GAN-tastisch: Eine Analyse zum „Image-to-Image Translation“-Framework „Pix2Pix“

Sebastian Mojado, mojadseb@students.zhaw.ch
ZHAW Zürcher Hochschule für angewandte Wissenschaften
„CAS Machine Intelligence“– Modul „Machine Learning“
Winterthur, 1. Mai 2017

***Abstract*—“Generative Adversarial Networks” kurz “GAN” sind Netze, die momentan sehr populär sind. Dieser Bericht stellt GAN und in Detail die GAN-Erweiterung namens „Pix2Pix“ vor: Ein „Image-to-Image Translation“-Framework mit Conditional Adversarial Nets. Dessen spezielle Eigenschaften z.B. seine flexible Einsetzbarkeit werden aufgezeigt.**

I. Die Faszination an GAN

Sowohl in Medien/Tech-Blogs als auch in der Wissenschaftsgemeinschaft (mehrere eingereichte Arbeiten per Woche [1]) sind GANs (Generative Adversarial Nets) und deren Erweiterungen sehr beliebt, weil sie vielversprechend sind und immer realistischere, synthetische Daten (vor allem Bilder) generieren können.

Sogar Experten wie Yann LeCun verleiten sie zu Aussagen zu GAN wie:

“This, and the variations that are now being proposed is the most interesting idea in the last 10 years in ML, in my opinion.”

Auch mich fasziniert dieser „Hype“. Die anderen zwei populären Deep-Learning-Techniken im Zusammenhang mit Bildern: „Image/Art-Style-Transfer“ und „Bild-Inhalt-Erraten“ sind auch interessant, aber GAN verstand ich am wenigsten und dies ist der Grund wieso ich mich für mein Wissen in GAN vertiefen wollte.

Dieser Bericht zu [2] (kurz „Pix2Pix“) zeigt einen interessanten Weg und eine Einfachheit (und dadurch Schönheit), die so sehr reizt, dass ich sie mit verschiedenen eigenen Daten anwenden begehrte. Wir befinden uns erst am Anfang der Möglichkeiten von GANs.

II. GAN – Die Grundlage

GAN gehen zurück auf [3] und können grob so zusammengefasst werden:

GANs bestehen aus zwei Netzen: einem Generator (G) und einem Diskriminator (D). Beide werden gleichzeitig trainiert und arbeiten gegeneinander in einem MiniMax-Spiel [4]. Der G versucht den D zu täuschen indem G echt-ausschauende Bilder generiert. D versucht sich nicht täuschen zu lassen. Zu Beginn erstellt G also Bilder durch das „Sampling“ eines Rausch-Vektors Z und dann durch das „Upsampling“ diesers. In den ersten Iterationen sieht das dann begrenzt realistisch aus. D vergleicht dann diese Bilder mit echten Bildern und lernt sie zu unterscheiden. G kriegt ein „Feedback“ von D durch einen Backpropagation-Schritt und kann so wiederum bessere Bilder generieren. Schlussendlich sollen die generierten Bilder so nahe wie möglich an den echten Bildern sein.

Da es bei GANs nun mind. zwei Netzwerke sind, die sich gegenseitig mit einem Min-Max-Spiel weitertreiben, ergeben sich verschiedenen Probleme:

- Verbesserungen/Konvergenz kann schwer gemessen werden anhand Trainingskurven.
- Stabilität: Ein Netzwerk kann sich wesentlich besser/schlechter als das andere verhalten. Insbesondere wenn der Diskriminator wesentlich besser ist als der Generator, produziert dieser immer schlechtere Bilder.
- Performance ist noch begrenzt, d.h. mir sind keine performante HD-Bilder-Anwendungen bekannt

Nichtdestotrotz funktionieren GANs erstaunlich gut und da diese Technik erst in den Kinderschuhen ist, ist zu erwarten, dass diese Probleme bald behoben werden. Eine Methode, die aufzeigt, wie verschiedene Bilderkategorien erzeugt werden können, ohne für jede Kategorie eine eigene Architektur oder „loss“-Funktion zu erstellen und trotzdem stabil bleibt ist die „Pix2Pix“-Methode.

III. “Pix2Pix” in Detail

Hauptthema dieses Reports ist die Arbeit [2] mit dem Namen „Image-to-Image Translation with Conditional Adversarial Networks“ von Isola, Zhu, Zhou und Efros von der UC Berkeley

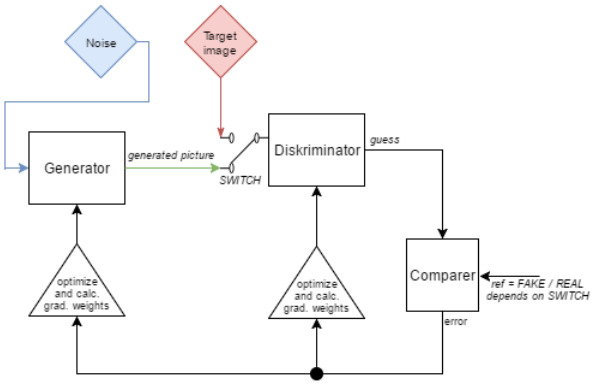

Fig 1. Standard-GAN-Training-Konzept

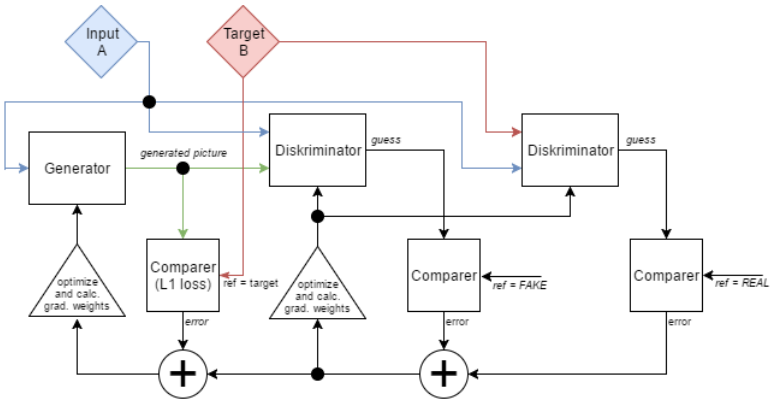

Fig 2. "Pix2Pix"-cGAN-Training-Konzept

(folgend genannt „die Autoren"). Allgemein und auch hier in diesem Bericht wird auf diese Arbeit referenziert mit dem Stichwort „Pix2Pix".

Wie der Name sagt, ist die Ziel-Anwendung, das „Übersetzen" eines Quell-Bildes in ein korrespondierendes Ziel-Bild. Zum Beispiel von einem Satellitenbild in eine stilisierte Karte, von einem Landschaftsbild bei Tage in das gleiche Bild aber als Nachtaufnahme oder im Winter. Richtig populär geworden ist das ganze durch weitere Portierungen und der Online-Übersetzung von eigenen Skizzen in entsprechende „Real"-Bilder [5].

So ein „Übersetzen" von solch verschiedenen Bilderkategorien benötigt bis anhin anwendungsspezifische Algorithmen, also spezifische Neuronale-Netzwerk-Architektur und „loss"-Funktion. Zwar lernen CNNs ihre „loss"-Funktion selber zu minimieren, aber es muss immer noch applikationsspezifisch und mit Expertenwissen formuliert werden, welche Funktion zu minimieren ist. Das eigentliche Hauptziel von „Pix2Pix" sind nicht lustige Katzen-Mutanten-Bilder [6], sondern ein generischer Vorgang, der dieses manuelle Anpassen weglässt und für alle Bilder-übersetzungen gilt. Ein „common framework" [2, Seite 1] sozusagen.

Um die Aufgabe zu meistern, ein allgemeingültiges Rezept anzubieten und gleichwohl eben scharfe und realitätsnahe Bilder zu generieren haben die Autoren sich den „Conditional Adversarial Networks" (cGAN) von [7], eine Erweiterung der GAN, bedient.

Ein cGAN benutzt Zusatzinformation („label information") um die Qualität der generierten Bilder zu steigen und ein Stück weit auch zu kontrollieren. Beispiele solcher Zusatz-information sind das Geschlecht der abgebildeten Person oder die Information, ob sie eine Brille trägt. Diese Zusatzinformationen werden sowohl in den Generator G als auch in den Diskriminator D eingespeist. Sie führen dazu, dass das Modell bessere und interessantere Bilder generiert. Zum Beispiel möchte man aus einem Text ein Bild generieren, aber nun zusätzlich Angaben machen, wo die gewünschten Objekte sich befinden [8].

Die „Pix2Pix"-Framework benutzt ein cGAN, verwendet aber keine klassischen „label-information", sondern das „Quell"-Bild selber. Das heisst, beide Netze (G+D) können neben den generierten Bildern auch die Quellbilder sehen. Bemerkenswert ist hierbei die Einfachheit, wie alle vorhanden Informationen verwendet werden, aber das Ganze nicht applikationsspezifisch ist, wie bei anderen cGAN-Abhandlungen.

Dieses Miteinbeziehen des Quellbildes macht die eigentliche Aufgabe, nämliche die der Bild-zu-Bild-Übersetzung erst möglich.

In *Fig 2.* sind die Unterschiede zu Standard-cGAN und Standard-GAN gut sichtbar:

- G und D können neben den generierten Bildern auch die Quellbilder sehen
- D lernt sowohl vom generierten Bildern (FAKE) als auch von Zielbild (REAL)
- Es wird ein L1 Kostenfunktion angewendet, die pixel-weise das generierte Bild bewertet. In Standard-GAN macht man das nicht, weil ja neuartige Bilder generiert werden sollen. Bei einer Bild-Übersetzung ist aber diese Genauigkeit wichtig.

Bemerkenswert sind auch:

- G benutzt eine Architektur namens „U-Net", bei welcher der Encoder und Decoder stufenweise ihre Verbindung überspringen können. So überlistet G einen Informations-flaschenhals und ist performanter. Dies macht bei einer Bild-Übersetzung eben durchaus Sinn, da gewisse Kanten oder

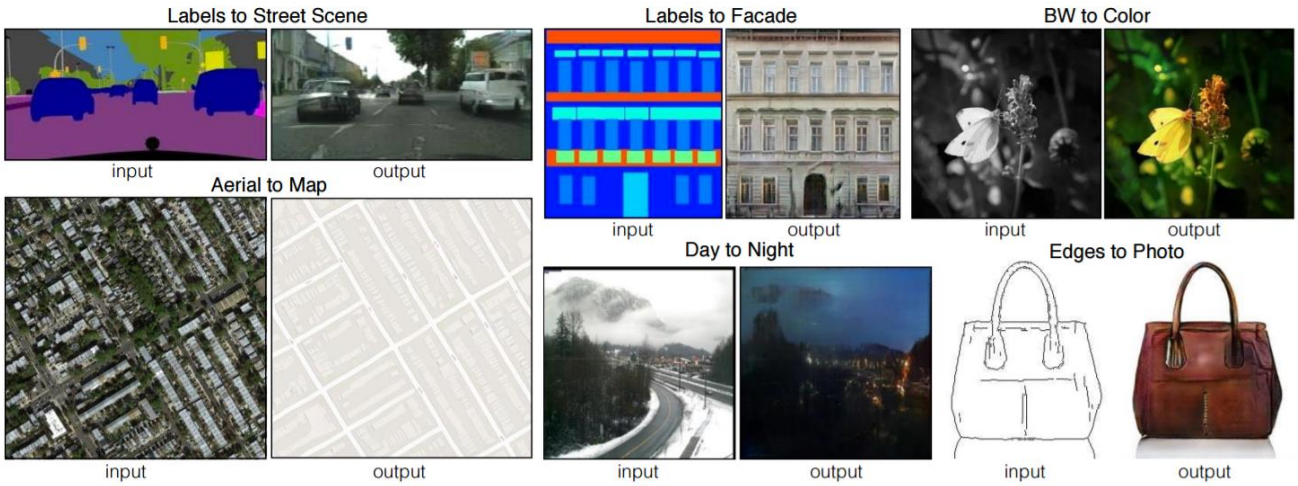

Fig 3. Diese Beispiele von den Autoren zeigen auf, wie vielfältig einsetzbar dieses „Framework“ ist

Farben ja immer an der gleichen Stelle bleiben.

- Bei „guess“ beim D in *Fig 2.* kommt kein boolscher Ausdruck raus, sondern ein 30x30x1 „Bild“, bei welchem jeder Pixel für die Glaubwürdigkeit einer Sektion (70x70 patch) steht. Die Autoren nennen dies PatchGAN.

IV. Pix2Pix in Anwendung

Da ja einer der Hauptmerkmale von „Pix2Pix“ die flexible Einsatzbarkeit ist, sind eigene Tests angebracht. Es gibt zwei populäre Github-Projekte: Die „offizielle“ Implementation [9] der Autoren in PyTorch und eine Tensorflow-Portierung [10] von Christopher Hesse. Dazu gibt es auch verschiedene hilfreiche Docker-Images [11]. Ich habe die Tensorflow-Variante benutzt und ebenfalls Bilder in der Dimension 256x256 Pixel verwendet.

A. *Versuch 1: Fülle den Inhalt rot*

Es wurden Bilder mit einem Oval generiert und dazu die gewünschte Zielversion erstellt mit einem kongruenten Oval aber mit roter Farbe ausgefüllt. Das Ziel war es also, eine Oval-Form rot auszufüllen zu lassen.

Fig 4. Beispiels-Trainingsbilder

Die Aufgabe war nicht besonders schwierig, aber bedenkt man, dass nur 400 Bilder (400 Trainingsbilder, 100 Testbilder) benutzt wurden, ist das Resultat doch erstaunlich:

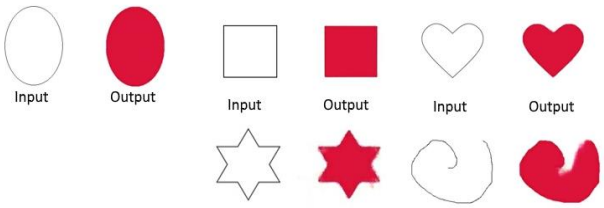

Fig 5. Validationsbilder mit „Red oval“-Datensatz

Auch „unbekannte Muster“ wie Quadrate oder Herzen werden ausgemalt. Erst bei ganz anderen total „trainings-fremden“ Formen zeigt sich grafische Unschönheiten (Pixel-Artefakte).

B. *Versuch 2: The Mojado-Face dataset*

Fig 6. Beispiels-Trainingsbild

Auch dieser Datensatz wurde selber erstellt. Er besteht aus ~600 Bilder mit einem Gesicht und dem entsprechenden Kantenbild. Ziel ist es anhand von einer Gesichts-Zeichnung, ein realitätsnahes Bild von dem Gesicht mit der entsprechenden Grimasse zu erstellen. Das Datenset ist auf Github [12] und wurde trainiert mit einer Azure NV6 mit 1xM60 für 5 Stunden.

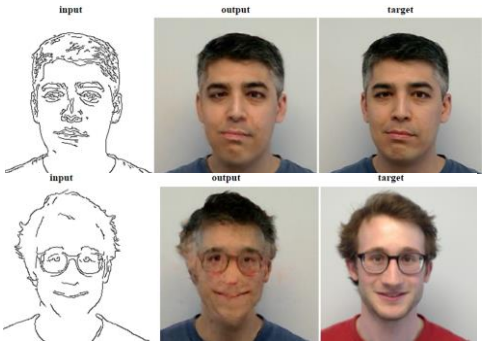

Fig 7. Die Resultate sind zwiespältig. Trainingsdaten-nahe Bilder sind sehr realitätsgetreu. Weicht man davon ab, werden die Bilder unnatürlich.

V. GANs und ihre Anwendung in Machine Learning

GANs werden in verschiedenen Gebieten verwendet: De-Blurisation von verpixeltem Video-Material, Stiländerungen von computer-generierten Designs (Zimmereinrichtungen, Möbel…), Rekonstruktion von 3D-Modellen aus Bildern, Bildvervollständigung, Verbesserung von Astronomiebildern, verbesserte und neue Bild-Editor-Programmfunktionen etc.

Auch „Pix2Pix“ selber wird weiterentwickelt wie in [13] und ist nach wie vor im Gespräch [14].

Zwar wurde bei der Recherche und bei den Experimenten mit „Pix2Pix“ festgestellt, dass wenn man nur ein wenig abweicht von den Trainingsdaten, die Bilder doch gekünstelt verzehrt, ja gar verschmiert werden.

Fig 8. Mojado-Face-Modell mit Skizze

Wir stehen aber erst am Anfang. GANs basieren auf einer simplen aber (oder vielleicht genau darum) mächtigen Idee: Zwei Neuronale Netze arbeiten gegeneinander, um so besser zu werden. Gute Ideen kann man meistens bei der Biologie abschauen. Ein Analogon wie von künstlichen neuronalen Netzen zu menschlichen Nervenzelle konnte aber bei dieser Arbeit zu „Adversarial Nets“ nicht gefunden werden. Am nächsten kommt die „Split-Brain“-Theorie [15] oder Marvin Minskys Vorstellung von Zensur- und Unterdrücker-Agenten [16].

Mit GANs sind Projekte wie [17] umgesetzt worden, die „in die Zukunft schauen“ können. Fürs Training wurden viele Videoframes eingelesen und in der Anwendung, den weiteren Verlauf (Frame für Frame) generiert. Der latente Raum soll also so Handlungen kennen und speichern können. Dies könnte ein nötiger Baustein sein für eine bessere Roboter-„motion planing“. Oder sogar Mustervorhersage wie in [18], was möglicherweise eine wichtige Basis für eine „artificial general intelligence“ sein kann.

GANs sind stark im Zusammenhang mit Bildern. Für Computerlinguistik oder Audio-Anwendungen produzieren RNN wesentlich bessere Resultate. Mögliche künftige GAN-Anwendungen sind:

- Computerspiele, die grafische Welten frisch generieren basierend auf der Stimmung des Spielers.
- GANs, die ihre gelernten Modelle teilen und kombinieren können. Ein GAN, dass z.B. lernt, was „Barbara Streisand“ ist und eines was ein „Roboter“ ist, um dann Varianten von „Robo-Streisand“ zu generieren.
- „Augmented Reality“-Applikationen wie Auto-Bildschirme, die den Schnee oder Regen wegzeichnen.
- Hollywood-Stars müssen keine Angst haben wegen Nacktbildern, weil die Menge an Nacktbildern, die generiert werden so gross ist, dass sie von echten nicht mehr unterschieden werden kann.

Im Zusammenhang mit GANs und „Style-Transfer“-Techniken wird auch oft das Thema Kreativität angesprochen. Viele befürchten, dass nun Maschinen menschlichen Künstler ersetzen. Ich bin der Meinung, dass das Gegenteil der Fall sein wird und kein Ersetzen stattfindet, sondern ein gegenseitiges Beflügeln [19]: Ein gesundes Konkurrenzdenken, ja fast eine „adversariale“ Beziehung zwischen Künstler und ML-Tool.

VI. Referenzen

[1] http://www.arxiv-sanity.com/search?q=gan

[2] *Phillip Isola, Jun-Yan, Zhu Tinghui, Zhou Alexei und A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks", Nov 2016,* https://arxiv.org/abs/1611.07004

[3] *Ian Goodfellow et al., "Generative Adversarial Nets", Juni 2016,* https://arxiv.org/abs/1406.2661

[4] https://de.wikipedia.org/wiki/Min-Max-Theorem

[5] https://affinelayer.com/pixsrv

[6] http://knowyourmeme.com/memes/sites/edges2cats

[7] *Mehdi Mirza und Simon Osindero, "Conditional Generative Adversarial Nets", Nov 2014, https://arxiv.org/abs/1411.1784*

[8] *Scott Reed et al., "Learning What and Where to Draw", Okt 2016,* https://arxiv.org/abs/1610.02454

[9] https://github.com/phillipi/pix2pix

[10] https://github.com/affinelayer/pix2pix-tensorflow

[11] https://hub.docker.com/search/?&q=pix2pix

[12] https://github.com/cBashTN/Pix2Pix_MojadoFaceDataSet

[13] *Jun-Yan Zhu, Taesung Park, Phillip Isola und Alexei A. Efros, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks", März 2017,* https://arxiv.org/abs/1703.10593

[14] https://twitter.com/search?&q=pix2pix

[15] http://www.cgpgrey.com/blog/you-are-two

[16] *Marvin Minsky ,"The Society of Mind: Mentopolis", 1994, Klett-Cotta-Verlag*

[17] *Jeff Hawkins und Sandra Blakeslee: On Intelligence. Owl Books, 2005*

[18] *Michael Mathieu, Camille Couprie und Yann LeCun, "Deep multi-scale video prediction beyond mean square error", Nov 2015,* https://arxiv.org/abs/1511.05440

[19] https://people.eecs.berkeley.edu/~junyanz/projects/gvm