

# Progressive Adversarial Neuronal Network for Image Creation

Jonas Wallat<sup>1</sup> und Josef Kriegel<sup>2</sup>

<sup>1</sup> Wichtiger Student der LuH, Welfengarten 1 30167, Deutschland,  
[jwallat@hotmail.com](mailto:jwallat@hotmail.com),

<sup>2</sup> Auch wichtiger Student der LuH, Welfengarten 1 30167, Deutschland,  
[josef.kriegel@web.de](mailto:josef.kriegel@web.de)

**Zusammenfassung.** Diese Arbeit führt in das Thema Generative Adversarial Networks (GANs), einer Technik zur Bilderzeugung mit neuronalen Netzen, ein. Der Fokus liegt darauf, zu lernen, wie mit neuronalen Netzen Bilder erzeugt werden können und welche Aufwände die Erzeugung mit sich bringen. Dabei wird sowohl auf die Entwicklung der Generative Adversarial Networks als auch auf die häufigen Probleme und Lösungsansätze eingegangen. Anschließend wird ein DCGAN verwendet, um mit einem kleinen medizinischen Datensatz künstliche Bilder zu erzeugen.

**Schlüsselwörter:** Generative Adversarial Networks, Medizin, GANs, Hacks

## 1 Einleitung

So, wie die Zahl der Veröffentlichungen zu Neuronalen Netzen in den letzten Jahren angestiegen ist, wächst auch die Community rund um eines der größten Forschungsbereiche des jungen 21. Jahrhunderts, das Machine Learning. Mit dem Wachstum der Community kommen viele neue und interessante Ideen auf, in welche Formen und Richtungen maschinelles Lernen noch getrieben werden kann. Im Jahr 2014 hat Dr. Goodfellow, seit März 2017 Mitarbeiter bei Google Brain und beteiligt an der Entwicklung von Deep Learning, eine neue Art von Neuronalen Netzwerken erfunden. Die sogenannten Generative Adversarial Networks (GANs). Die simple Theorie dahinter ist, dass zwei Netzwerke auf einem nicht klassifizierten Datensatz trainieren können und in der Lage sind, nach angemessenem Training, Bilder zu erzeugen, welche dem trainierten Datensatz soweit ähnlich sehen, dass es Menschen kaum noch möglich ist, einen Unterschied zwischen echten und generierten Bildern auszumachen. Das eine Netzwerk, genannt Generator, erzeugt aus zufällig generiertem Rauschen ein Bild, welches das zweite Netzwerk, der Diskriminatator, zusammen mit dem echten Datensatz als echt oder falsch klassifizieren muss. Anhand der Aussage des Diskriminators lernt der Generator dann, wie er die Bilder noch realistischer, genauer gesagt noch ähnlicher dem echten Datensatz, erzeugen kann.

GANs können also zur Erzeugung künstlicher Bilder verwendet werden, aber die Theorie wurde noch weiter getrieben. Auf dem Prinzip der GANs beruhend existieren Ansätze zur Generierung von Gesichtern, Musik, 3D Objekten oder auch zur Vorhersage des nächsten Bildes in einem Video [1]. Vorstellbar wäre, dass GANs in Zukunft unterstützend bei kreativen Prozessen und prozedural generiertem Inhalt für beispielsweise Computer-Spiele verwendet werden können. Was bei so rasant fortschreitender Technologie aber ebenfalls betrachtet werden sollte, ist die Gefahr des Missbrauchs. Wenn Bilder oder vielleicht auch Stimmen künstlich erzeugt werden, lassen sich GANs auch zur Täuschung von Systemen und Gesellschaft genutzt werden.

## 2 Verwandte Arbeiten

Seit der ersten Arbeit von Ian Goodfellow et al bezüglich GANs sind viele weitere Veröffentlichungen zum Thema GANs erschienen. Unter anderem weitere Verwendungen von Improved GANs, die Weiterentwicklungen DCGANs, Conditional GANs oder auch Least Squared GANs.

Eine beispielhafte Verwendung von GANs und DCGANs präsentiert die Arbeit Improved Techniques for Training GANs [2]. Visuell gegenübergestellt werden DCGANs, welche semi-supervised, und DCGANs, welche mit minibatch discrimination lernen. Minibatch discrimination bedeutet, dass der Diskriminator beim Evaluieren eines Bildes außerdem die Ähnlichkeit zu anderen Samples vom Generator misst und die Ähnlichkeit bei der Entscheidung, ob das Bild vom Generator kommt oder echt ist, mit einbezieht. Die Ergebnisse der Gegeüberstellungen wurden unter Anderem mit dem MNIST und dem ImageNet Datensatz evaluiert und sind in den Abbildungen 1 und 2 dargestellt.



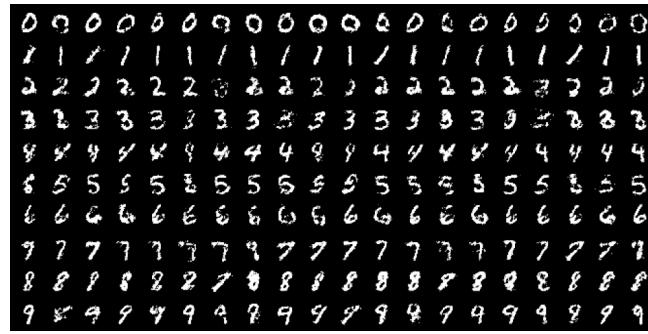
**Abb. 1.** MNIST Bilder generiert mit semi-supervised learning (links) und mit minibatch discrimination (rechts)[2]



**Abb. 2.** Hunde-Bilder generiert mit semi-supervised learning (links) und mit minibatch discrimination (rechts)[2]

Der Output des DCGAN in Abbildung 2 links hat zwar die Features gelernt, schafft es jedoch nicht wirkliche Hunde abzubilden. Das improved DCGAN rechts erzeugt bessere Bilder, die zumindest die Charakteristiken von Hunden abbilden, auch wenn die Anatomie nicht stimmt [2].

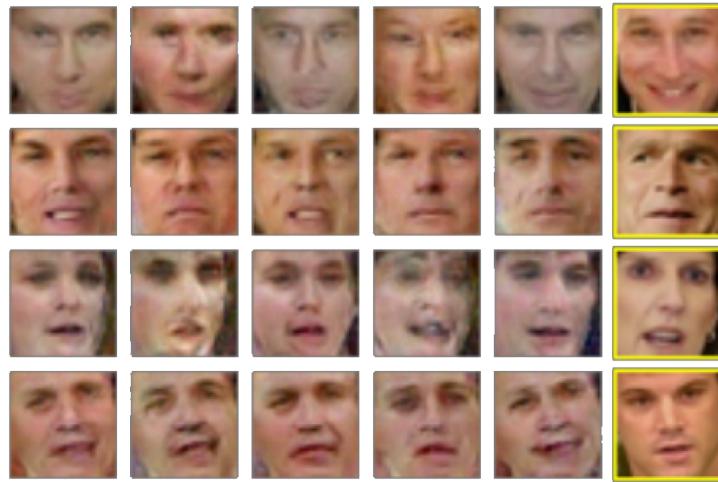
Eine weitere Arbeit stammt von Mehdi Mirza und Simon Osindero[3] und behandelt die Conditional GANs. Conditional GANs stellen eine Erweiterung von gewöhnlichen GANs um eine weitere Input Layer sowohl beim Generator als auch beim Diskriminatoren. Beiden Netzen werden also weitere Informationen pro Sample hinzugefügt. Dies kann beispielsweise ein Klassenlabel sein, ist jedoch nicht darauf beschränkt. Mirza et al haben ersteres umgesetzt und ein Model mit dem MNIST Datensatz trainiert, welches ein Klassenlabel als Input erhält. Mit diesem Input lässt sich vorherbestimmen, welcher Klasse das erzeugte Sample des Generators entspricht. Beim MNIST Datensatz entspricht eine Klasse der Zuordnung zu den Zahlen 0 bis 9. Die Ergebnisse sind in Abbildung 3 dargestellt.



**Abb. 3.** MNIST Bilder generiert mit einem Conditional GAN[3]

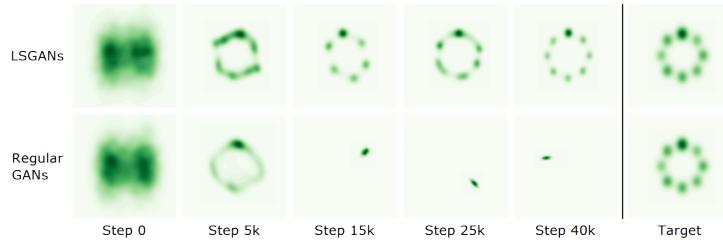
Auch Gesichter können mit Conditional GANs gezielter erzeugt werden. Das zeigt die Arbeit Conditional generative adversarial nets for convolutional face

generation von Jon Gauthier [4] mit dem Datensatz Labeled faces in the wild (Wild Datensatz). Der konditionelle Aspekt wird hier genutzt, um bestimmte Merkmale beim Output des Generators zu erzwingen. Genauer gesagt wird aus den elementaren Attributen der originalen Gesichter, 73 an der Zahl, eine Teilmenge von 36 Attributen extrahiert und als weitere Input Layer dem zugrundeliegenden GAN hinzugefügt. Dies entspricht stark abstrahiert beispielsweise der Kondition, dass das Ausgangsbild zwei Augen, eine Nase und einen Mund enthalten soll. Ein Teil der visuellen Ergebnisse der Arbeit finden sich in Abbildung 4 wieder.



**Abb. 4.** Gesichts-Bilder generiert mit einem Conditional GAN[4]

Als letzte verwandte Arbeit soll die Veröffentlichung Least Squares Generative Adversarial Networks von Xudong Mao et al[5] erwähnt werden. Die hauptsächliche These hinter Least Squares GANs stellt der negative Einfluss der standardmäßigen sigmoid cross entropy Loss Funktion auf die Schrittweite der Gewichtsveränderung im Laufe des Trainingsprozesses dar. Das Problem soll mit abgewandelten Loss Funktionen für Generator und Diskriminatator umgangen werden. Wie sich die neuen Loss Funktionen auf ein nicht näher beschriebenen normalverteilten Datensatz auswirkt, kann in Abbildung 5 beobachtet werden. Interessant ist, dass reguläre GANs circa ab Epoche 15k nur noch valide Punkte in einem einzelnen gültigen Bereich der Verteilung erzeugen, wohingegen LSGANs die Mischung der Normalverteilungen gelernt haben [5]. Das Ergebnis der Anwendung von LSGANs auf die Datensätze LSUN, CIFAR-10 und HWDB1.0 bestätigt die Theorie, dass LSGANs deutlich hochwertigere Bilder erzeugen können, als es reguläre GANs vermögen (siehe [5]).

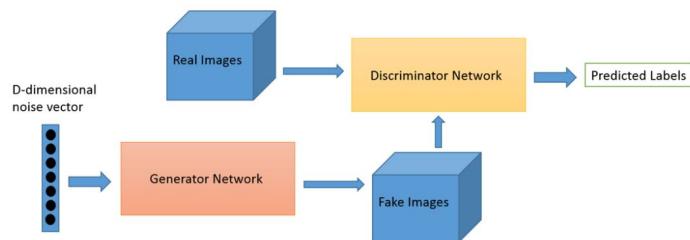


**Abb. 5.** Dynamische Ergebnisse des Gauss-Kernel-Schätzung für LSGANs (oben) und reguläre GANs (unten)[5]

### 3 Generative Adversarial Networks

In diesem Abschnitt wird zuerst in den ersten Vorschlag für Generative Adversarial Networks, wie von Goodfellow et. al. [6] vorgeschlagen, eingeführt. Danach wird aus den aufgetretenen Problemen die Entwicklung der nachgegangenen Forschung erläutert und auf Praxis-nahe Tricks eingegangen, die bei im Training auftretenden Problemen helfen können. Zuletzt wird ein Ausblick gegeben, wofür die ausgefeilteren Ansätze verwendet werden können und woran aktuell geforscht wird.

Wie der Name Generative Adversarial Networks bereits nahelegt, werden zwei gegenüberstehende neuronale Netze verwendet, um in diesem Fall Bilder zu erzeugen. Die gegeneinander arbeitenden Netze sind der Generator und der Diskriminatator. Der Diskriminatator ist dabei ein einfaches neuronales Netz zur Klassifizierung von Bildern als entweder echt oder falsch. Echt ist ein solches Bild, wenn es aus den Trainingsdaten stammt und falsch, wenn es vom Generator erzeugt wurde. Das Ziel des Diskriminators ist es, möglichst genau zwischen echten und falschen Bildern unterscheiden zu können. Im Generator wird aus einem Noise-Vektor und den gelernten Gewichten ein Bild erzeugt wird. Das Ziel ist, so echt wirkende Bilder zu erzeugen, dass der Diskriminatator diese nicht von aus dem Trainingsdatensatz stammenden Bildern unterscheiden kann. Der Generator lernt, indem für jedes erzeugte Bild die Klassifizierung des Diskriminators an den Generator zurückpropagiert und die Gewichte angepasst werden.



**Abb. 6.** Vereinfachte Architektur eines GANs[7]

Interessant an diesem Ansatz ist es, die zwei Netze gegeneinander zu trainieren. Auf diese Weise werden sich beide Netze immer weiterentwickeln und insgesamt bessere Ergebnisse erzielen. Umgesetzt durch das Einbinden des Diskriminators in die Optimierung des Generators ergeben sich die folgenden Gradienten:  
Diskriminator:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m [\log D(x^{(i)}) + \log(1 - D(G(z^{(i)})))] \quad (1)$$

Besonders relevant ist zum Verständnis des Gradienten lediglich der Inhalt der Summe: Der Gradient setzt sich aus dem Ergebnis des Diskriminators auf echten Bildern ( $D(x)$ ) und dem Ergebnis auf vom Generator aus dem Noise-Vektor erzeugten, falschen, Bildern ( $D(G(z))$ ) zusammen. Da wir für alle falschen Bilder eine Ausgabe 0/falsch anstreben, wird im Gradienten  $1 - D(G(z))$  verwendet. Für den Generator ergibt sich:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \log(1 - D(G(z^{(i)}))) \quad (2)$$

Analog zum vorherigen Gradienten betrachten wir erneut den Inhalt der Summe: Der Gradient des Generators besteht aus  $1 - \text{der Vorhersage des Diskriminators}$  bei Eingabe generierter Bilder, denn der Generator soll dem Diskriminator entgegenarbeiten und versuchen, ihn zu täuschen [6].

Diese verwobenen Trainingsfunktionen ergeben ein Konstrukt in dem die beiden neuronalen Netze in ihrem Lernen von dem erfolgreichen Lernen des anderen Netzes abhängen, denn nur so können sich beide Netze stetig verbessern. Der Generator wird gefordert immer echter wirkende Bilder zu erzeugen, die der Diskriminatior nicht mehr von echten unterscheiden kann und der Diskriminatior wird gefordert, die echten Bilder immer besser von falschen unterscheiden zu können.

Generative Adversarial Networks sind jedoch schwer zu trainieren. Dadurch, dass wir zwei neuronale Netze parallel trainieren, kommt es schnell zu Fehlerzuständen wie, dass der Diskriminatior schneller lernt und dem Generator keinen Verbesserungsspielraum lässt. Bekommt der Generator nur die Rückmeldung, dass alle erzeugten Bilder als falsch erkannt werden, lässt sich kein Verhalten lernen, das echt-wirkende Bilder erzeugt. Andererseits ist es auch möglich, dass der Generator Fehler des Diskriminators ausnutzt und viele sehr ähnliche Bilder produziert, obwohl sich der Input ändert. Das ist meist ein Zeichen für einen zu schwachen Diskriminatior und kann teilweise durch Nachtrainieren des Diskriminators behoben werden [7] [6].

### 3.1 Entwicklung der Generative Adversarial Networks

Die Idee der Generative Adversarial Networks stößt bei der Veröffentlichung 2014 von Goodfellow und Kollegen auf großes Interesse der Machine Learning Com-

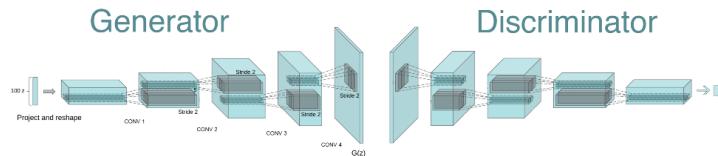
munity. So zum Beispiel Yann Lecun (Direktor bei Facebook AI und Professor an der New York University)[8]:

“Adversarial training is the coolest thing since sliced bread“

Die Ursprüngliche Variante der Generative Adversarial Networks hat allerdings einige Probleme, die die Verwendung auf großen und komplexeren Datensätzen nur sehr bedingt möglich macht. Zum einen, dass häufiges Fehlschlagen des Trainings und Auftreten der oben beschriebenen Fehlerzustände und wahrscheinlich ist und zum anderen, dass eine sehr hohe Zahl an Epochen notwendig, um akzeptable Ergebnisse zu erhalten. Hinzu kommt, dass die Ergebnisse beim Hyperparameter Tuning nur schlecht verständlich sind. Letztlich erschweren wenig aussagekräftige Loss Funktionen die Interpretation des Trainingsfortschritts und die Entscheidung, wann mit dem Trainieren aufgehört werden kann. Seit 2014 haben sich jedoch einige Forschungsarbeiten mit den Ideen Goodfellow und ihren Problemen auseinandersetzt und diese weiterentwickelt. Eine kurze chronologische Darstellung der für diese Arbeit wichtigsten Forschung folgt.

### Deep Convolutional Generative Adversarial Networks

Die erste Weiterentwicklung der GANs stellen die Deep Convolutional Generative Adversarial Networks (DCGANs) dar. Diese zeichnen sich vor allem durch eine überarbeitete Netzwerkarchitektur und Hyperparameter-Tuning aus. Der Diskriminatator wurde durch ein Convolutional Neuronal Network und der Generator durch ein Deconvolutional Neuronal Network ersetzt (siehe Abbildung 7).



**Abb. 7.** Generator und Diskriminatator als Convolutional Neural Networks [9]

Durch diese Änderung konnte verlässlicher trainiert und hochwertigere Bilder erzeugt werden. Weitere eingebrachte Vorschläge waren die Verwendung von batch normalization, ReLU Aktivierungsfunktionen sowohl im Generator, als auch im Diskriminatator, das Vermeiden von fully hidden connected layers und pooling [10]. DCGANs bringen weit bessere Ergebnisse als ursprüngliche GANs und werden häufig noch als baseline für andere GANs verwendet [11].

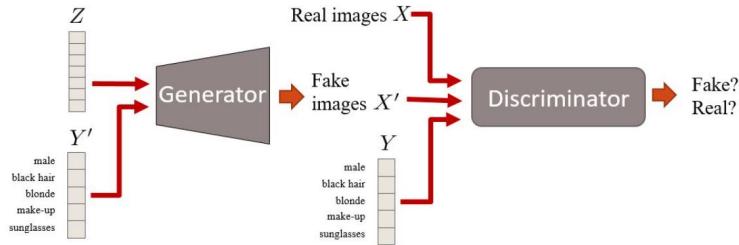
### Improved Deep Convolutional Generative Adversarial Networks

Die Improved Deep Convolutional Generative Adversarial Networks (Improved DCGANs) sind eine Weiterentwicklung der vorangegangenen DCGANs und erlauben durch kleinere Änderungen am Training das Erzeugen von höher aufgelösten Bildern und ein stabileres Training. Dazu wird eine neue objective

function eingeführt, in der das Ziel des Generators nicht mehr hauptsächlich das Täuschen des Diskriminators ist, sondern die Charakteristiken der Trainingsdaten möglichst gut abzubilden. Weiterhin werden beim Aktualisieren der Parameter die vorherigen Werte des Parameters miteinbezogen. Auch wurden die Ergebnisse verbessert, indem one-sided label smoothing betrieben wird. Das heißt, dass die Ausgabe des Diskriminators von 0/1 bei falschen/echten Bildern zu 0/0.9 geändert wurde [2].

### Conditional Generative Adversarial Networks

Conditional Generative Adversarial Networks (cGANs) erweitern vorherige Modelle um Attribute der Motive in den Bildern. Auf diese Art und Weise könnte diese Attribute gelernt und beim Erzeugen neuer Bilder bedacht werden. CGANs erlauben also eine gewisse Kontrolle über den Output des Generators. Die schematische Umsetzung eines cGANs ist in Abbildung 8 zu sehen. Interessant ist, dass der Attribut-Vektor Y (bzw. Y') eine Eingabe für den Generator als auch den Diskriminator ist [3].

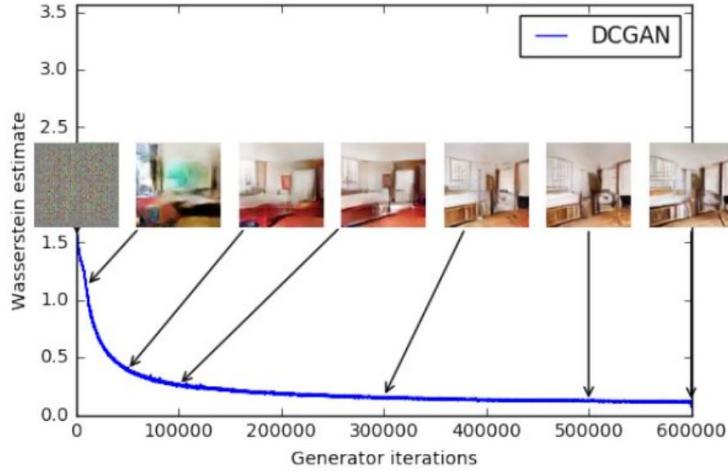


**Abb. 8.** Architektur eines Conditional Generative Adversarial Networks [11]

### Wasserstein Generative Adversarial Networks

Wasserstein Generative Adversarial Networks (WGANs) sind die letzte hier ausführlich behandelte Weiterentwicklung der ursprünglichen GANs und sind Anfang 2017 publiziert worden. Die Idee der Wasserstein GANs ist es, durch Einbinden der Wasserstein Distanz in die Loss Funktion die Qualität der erzeugten Bilder an die Loss Funktion zu binden. Dadurch können die Ergebnisse besser interpretiert und die Entscheidung wann das Training ausreicht vereinfacht werden. Eine Darstellung der Loss Funktion eines WGAN und wie diese an die Bildqualität gebunden ist sieht man in Abbildung 9.

Wasserstein GANs haben also den Vorteil einer gut interpretierbaren Loss Funktion. Durch die Verwendung der Wasserstein Distanz wurde ebenfalls die Trainingsstabilität verbessert [12].



**Abb. 9.** Loss Funktion eines WGAN gekoppelt an die Bildqualität [11]

### 3.2 GAN Tipps

Auch wenn viel an potentiellen neuen Architekturen und Verbesserungen für GANs geforscht wird, ist diese Forschung häufig noch sehr theoretisch. An dieser Stelle folgt eine kleine Zusammenstellung von Tipps und Tricks, die die Konvergenz von aktuell verwendeten GANs verbessern können. Die Zusammenstellung basiert auf [13]:

- Normalisierung des Inputs:  
Normalisierung des Bilder zwischen -1 und 1. Weiterhin ist es sinnvoll im letzten Layer des Generators den Tangens hyperbolicus zu verwenden
- Modifizierte Loss Funktion:  
Anstelle den Generator mit  $\min(\log(1-D))$  besser mit  $\max \log D$  optimieren
- Sampling:  
Zum Sampling anstelle Gleichverteilung lieber Normalverteilung wählen
- Batch Normalization:  
Nur Batches aus entweder echten oder generierten Bildern in den Diskriminator geben
- Soft und Noisy Labels:  
Die Label der echten Bilder mit einer zufälligen Zahl zwischen 0.7 und 1.2, sowie die Label der generierten Bilder mit einer Zahl zwischen 0.0 und 0.3 ersetzen
- Adam Optimizer:  
Für den Generator bietet der ADAM Optimizer die besten Ergebnisse
- Noise im Input:  
Ein Hinzufügen von Noise zum Input, der nach einiger Zeit geringer wird verbessert die Ergebnisse

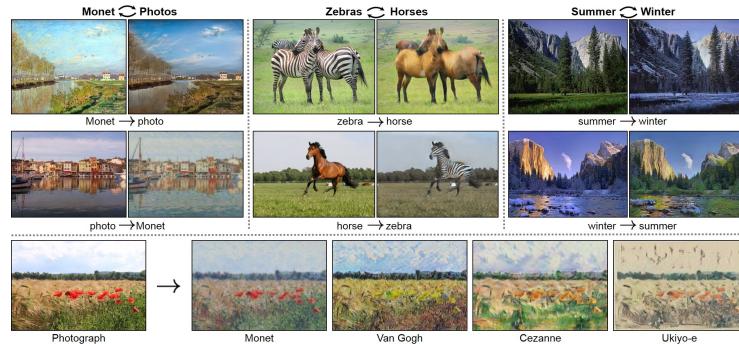
### 3.3 GAN Ausblick

Auch wenn die Forschung im Bereich der Generative Adversarial Networks in den letzten vier Jahren große Fortschritte gemacht hat, ist das Thema auch für zukünftige Forschung weiter interessant und wird weiter untersucht. Speziell die Frage, wie hochauflösende Bilder generiert werden können, steht zum mindesten teilweise aus - auch wenn in [14] bereits erste Erfolge erreicht werden konnten (siehe Abbildung 10). Dort wurden Bilder der Auflösung 1024x1024 erzeugt, höhere Auflösungen wie 2k oder 4k bereiten jedoch noch Probleme. Bisherige Beispiele bezogen sich auf meist auf Bilder der Größe 256x256.



**Abb. 10.** Generierte Bilder mit der Auflösung 1024x1024 [14]

Weiterhin wird aber auch mit der Anwendung von GANs in weiteren Bereichen experimentiert. So wurde in [15] gezeigt, dass Generative Adversarial Networks ebenfalls für style-transfer verwendet werden können (Abbildung 11).



**Abb. 11.** Beispiele für Style Transfer mit CycleGANs [15]

Insgesamt sind in diesem jungen Forschungsbereich noch viele Verbesserungen bestehender Ansätze und viele neue Verwendungsmöglichkeiten zu erwarten.

## 4 Projekt PANIC

Vielfache Variationen von GANs wurden bereits auf Datensätzen wie beispielsweise LSUN Bedrooms oder MNIST angewendet und publiziert. In dem Projekt

Progressive Adversarial (neural) Network for Image Construction (PANIC) soll an dieser Stelle ein GAN mit dem Framework Tensorflow mit einem kleinen Datensatz von 42 Bildern trainiert und anschließend zur Bildererzeugung genutzt werden. Der Datensatz besteht aus ausgeschnittenen einzelnen Zellen aus Elektronenmikroskop-Aufnahmen.

Die ausgeschnittenen Bilder haben flexible Größen zwischen ca. 50x50 und 100x100 Pixeln, sodass einige Schritte des Preprocessings notwendig sind. Um die Bilder in einen für Tensorflow kompatiblen zufälligen Batch zu konvertieren, wurde der Inhalt jedes Bildes zentriert und in ein 100x100 Pixel großes Bild eingefügt. Das Hochskalieren mit Verzerrung zu 100x100 Pixel wurde ebenfalls getestet, aber für ungeeignet empfunden, da es die Form der Zellen zu stark verfälscht.

Nach dem Laden der Bilder in Tensorflow werden diese zufällig horizontal gespiegelt, zufällig auf- oder abgehellt und zufällig kontrastiert. Dies dient der Vergrößerung der Varianz unter den Batches, sodass das GAN mit geringerer Wahrscheinlichkeit in geläufige lokale Maxima und Fehlerzustände läuft. Zuletzt werden die Farbwerte jedes Pixels von 0-255 auf 0 bis 1 normalisiert.

Anschließend wurden das diskriminative und das generative Netz definiert. Das diskriminative Netz, also das Netz, welches Bilder erhält und ausgeben soll, ob diese echt sind oder vom generativen Netz erzeugt wurden, besteht aus vier Layern. Die ersten beiden sind Convolutional Layer mit anschließendem Average Pooling, um die Dimensionalität zu reduzieren. Die letzten beiden Layer sind Fully Connected Layer bis hin zum eindimensionalen Ausgang, welcher die Wahrscheinlichkeit wiederspiegelt, mit der der Input ein echtes, oder ein generiertes Bild ist.

Das generative Netz besteht aus einem Layer, in welchem der zufällige Input projiziert und umgeformt wird. Danach folgen drei Convolutional Layer, welche aus vielen Features Layer für Layer den dreidimensionalen Raum immer weiter verkürzen und letztendlich bei einem großen zweidimensionalen Output, einem Bild mit einem Feature, enden. Eine Beispielhafte Darstellung der beiden Netzerwerke und ihres Zusammenspiels lässt sich in Abbildung 7 finden.

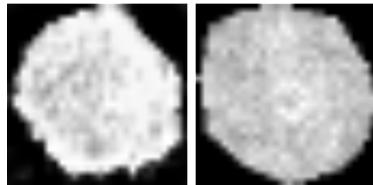
Zur Auswertung des Trainingsfortschritts wurde eine Visualisierung mit Tensorboard verwendet. In insgesamt 2000 Epochen wird alle 10 Epochen ein Bild vom Generator erzeugt. Außerdem wird der aktuelle Loss von Generator, also der relative Anteil an Bildern, die der Generator erzeugt und bei denen der Diskriminatator erkannt hat, dass sie nicht echt sind, angezeigt. Für den Loss des Diskriminativen Netzes existieren zwei verschiedene Loss Werte. Einerseits die Verlustrate für die echten Bilder und andererseits für die künstlich erzeugten Bilder.

Um der Gefahr, in die typischen Fehlerzustände (beschrieben in Kapitel 3), zu welchen GANs während des Trainings tendieren, zu laufen, entgegenzuwirken, wurde das Training von Generator und Diskriminatator konditioniert. Der Diskriminatator wird auf die Erkennung von gefälschten Bildern, falls die Erkennungs-

rate derselben kleiner als 60% ist, auf die Erkennung von echten Bildern, falls die Erkennungsrate dafür kleiner als 45% ist und auf die Erzeugung von Bildern des Generators, falls die Erkennungsrate der falschen Bilder vom Diskriminatorkleiner als 50% ist, trainiert.

## 5 Auswertung

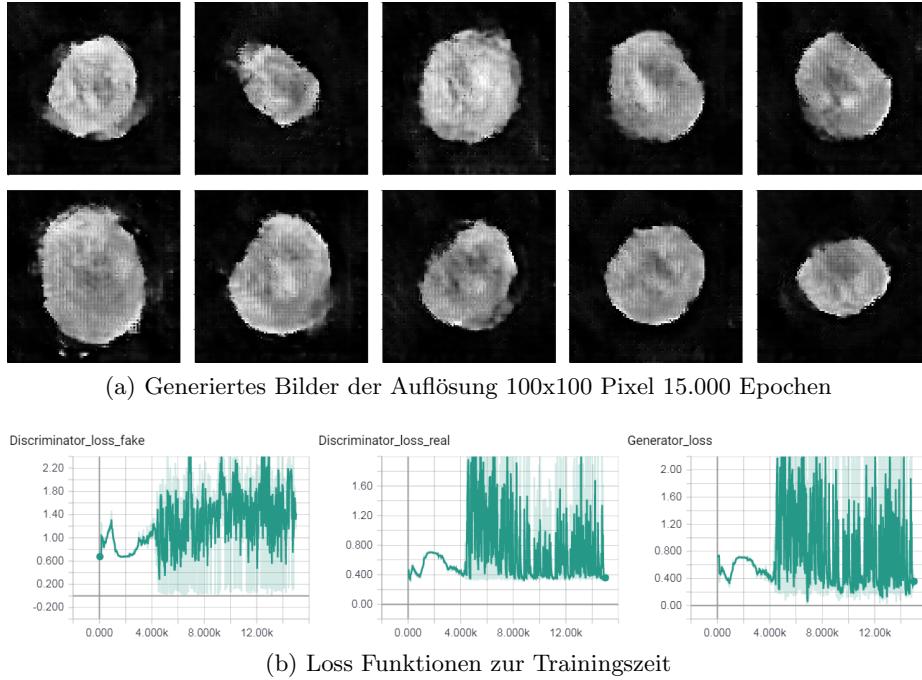
An dieser Stelle eine kurze Auswertung der Ergebnisse des Project PANIC. Bei Runterskalierung der Eingabebilder auf 28x28 Pixel liefert unser einfaches DC-GAN bereits nach 1.000 Epochen sehr gute Ergebnisse (Abbildung 12)



**Abb. 12.** Generiertes Bild (links) und Eingabe-Bild (rechts)

Nun wurde versucht, mit einem ähnlichen Netz eine Ausgangsauflösung von 100x100 Pixeln zu erreichen. Dies gestaltete sich jedoch schwieriger als erwartet, sodass die anfänglichen Versuche strukturell schlechtere Ergebnisse lieferten, als die Bilder in Abbildung 12. Mit etwas explorativer Testarbeit stellte sich heraus, dass leicht modifizierte Kernel-Größen in Convolutional Layers, sowie Batch-Normalisierung im Diskriminatorken die Qualität erheblich verbessert haben. Die verbesserten generierten Bilder in der Auflösung 100x100 Pixel sind in Abbildung 13 zu sehen. Anzumerken ist auch, dass an dieser Stelle mit 15.000 Epochen wesentlich mehr trainiert wurde, als im Falle der 28x28 Pixel Bilder und das Ergebnis dennoch nicht nur leicht besser ist. Erstaunlich war bei dem langen Training der Verlauf der Loss Funktionen. Der adversielle Charakter lässt sich gut erkennen.

Dass die vergrößerte Variante des DCGANs deutlicher mehr Training benötigte, ist wohl der höheren Auflösung sowie der Struktur des GANs geschuldet. Mit der fast sechzehnfachen Anzahl Pixeln pro Bild werden auch die sechzehnfache Anzahl Gewichte trainiert. Eine Änderung der Architektur oder das Verwenden eines Wasserstein GANs würde die Ergebnisse vermutlich noch weiter optimieren, wurde jedoch noch nicht getestet. Weiterhin ist die Loss Funktion, wie für DCGANs üblich, nicht besonders verständlich und macht wenig Aussage über die wirkliche Qualität des generierten Bildes, weshalb sich die Analyse, an welcher Stelle der qualitative Engpass liegt, sich als besonders schwierig gestaltet. Die bisherige Implementierung erfüllt allerdings das Forschungsziel zu verstehen, wie Generative Adversarial Networks funktionieren und ist in der Lage, mit ge-



**Abb. 13.** Trainingsergebnisse mit 100x100 Pixel Generat

wissem Trainingsaufwand auf dem relativ simplen Eingabedatensatz akzeptable Ergebnisse zu erreichen.

## 6 Ausblick

In Zukunft könnte das PANIC-Projekt auf ein Wasserstein GAN umgestellt werden, um die Loss Funktion nachvollziehbarer zu gestalten und die Trainingsstabilität zu verbessern. Des Weiteren müsste die Architektur angepasst werden, um gute Ergebnisse auch auf höher aufgelösten Bildern zu erzielen.

Ebenfalls könnte tiefer in die aktuelle Forschung, die in Kapitel 3.3 nur kurz erwähnt wurde, eingegangen werden. Für das Ziel, eine Einführung in Generative Adversarial Networks zu geben, ist das jedoch nicht notwendig. Peter

## Literatur

- [1] skymind. *A Beginner’s Guide to Generative Adversarial Networks (GANs)*. 2017. URL: <https://deeplearning4j.org/generative-adversarial-network> (besucht am 07.01.2018).
- [2] Tim Salimans u. a. „Improved Techniques for Training GANs“. In: *CoRR* abs/1606.03498 (2016). arXiv: 1606 . 03498. URL: <http://arxiv.org/abs/1606.03498>.

- [3] Mehdi Mirza und Simon Osindero. „Conditional Generative Adversarial Nets“. In: *CoRR* abs/1411.1784 (2014). arXiv: 1411 . 1784. URL: <http://arxiv.org/abs/1411.1784>.
- [4] Jon Gauthier. „Conditional generative adversarial nets for convolutional face generation“. In: *Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester 2014.5* (2014), S. 2.
- [5] Xudong Mao u. a. „Least squares generative adversarial networks“. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE. 2017, S. 2813–2821.
- [6] Ian Goodfellow u. a. „Generative adversarial nets“. In: *Advances in neural information processing systems*. 2014, S. 2672–2680.
- [7] Adit Deshpande und Jon Bruner. *Generative Adversarial Networks for Beginners*. 2017. URL: <https://www.oreilly.com/learning/generative-adversarial-networks-for-beginners> (besucht am 02.01.2018).
- [8] Yann Lecun. *What are some recent and potentially upcoming breakthroughs in unsupervised learning*. 2016. URL: <https://www.quora.com/What-are-some-recent-and-potentially-upcoming-breakthroughs-in-unsupervised-learning> (besucht am 06.01.2018).
- [9] Zachary C. Lipton. *Deep Convolutional Generative Adversarial Networks*. 2017. URL: [http://gluon.mxnet.io/chapter14\\_generative-adversarial-networks/dcgan.html](http://gluon.mxnet.io/chapter14_generative-adversarial-networks/dcgan.html) (besucht am 03.01.2018).
- [10] Alec Radford, Luke Metz und Soumith Chintala. „Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks“. In: *CoRR* abs/1511.06434 (2015). arXiv: 1511 . 06434. URL: <http://arxiv.org/abs/1511.06434>.
- [11] Guim Perarnau. *Fantastic GANs and where to find them*. 2017. URL: <http://guimperarnau.com/blog/2017/03/Fantastic-GANs-and-where-to-find-them> (besucht am 03.01.2018).
- [12] Martin Arjovsky, Soumith Chintala und Léon Bottou. „Wasserstein Generative Adversarial Networks“. In: *Proceedings of the 34th International Conference on Machine Learning*. Hrsg. von Doina Precup und Yee Whye Teh. Bd. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, 2017, S. 214–223. URL: <http://proceedings.mlr.press/v70/arjovsky17a.html>.
- [13] Soumith Chintala u. a. *GAN Hacks*. 2016. URL: <https://github.com/soumith/ganhacks> (besucht am 04.01.2018).
- [14] Tero Karras u. a. „Progressive Growing of GANs for Improved Quality, Stability, and Variation“. In: *CoRR* abs/1710.10196 (2017). arXiv: 1710 . 10196. URL: <http://arxiv.org/abs/1710.10196>.
- [15] Jun-Yan Zhu u. a. „Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks“. In: *CoRR* abs/1703.10593 (2017). arXiv: 1703 . 10593. URL: <http://arxiv.org/abs/1703.10593>.