

**Fachbereich 03: Rechts- und Wirtschaftswissenschaften**

**Lehrstuhl für Wirtschaftsinformatik**

**Prof. Dr. Franz Rothlauf**

**Sommersemester 2021**

Abgabe zum Thema

**DCGAN - Generieren von Gesichtern**

zur Veranstaltung Intelligent Information Systems

Abgabetermin: 16.07.2021

**Christian Loor**

Wallstraße 8  
55122 Mainz  
2743965  
[cloor@students.uni-mainz.de](mailto:cloor@students.uni-mainz.de)  
1. Fachsemester  
Management

**Kevin Kammler**

Schlesienstraße 5  
55595 Hargesheim  
2708347  
[kkammler@students.uni-mainz.de](mailto:kkammler@students.uni-mainz.de)  
3. Fachsemester  
Wirtschaftswissenschaft-  
liche Informatik

**Tomislav Pree**

Oggersheimerstraße 3  
67112 Mutterstadt  
2738523  
[tpree@students.uni-mainz.de](mailto:tpree@students.uni-mainz.de)  
2. Fachsemester  
Wirtschaftswissenschaft-  
liche Informatik

## Inhaltsverzeichnis

<b>Einleitung</b>	<b>1</b>
<b>Datensätze</b>	<b>1</b>
<b>Architektur und Implementierung</b>	<b>2</b>
GAN	3
DCGAN	4
GAN vs. DCGAN	5
Implementierung und Parameter	5
<b>Ergebnisse</b>	<b>6</b>
CelebA	6
Anime	8
Simpsons	8
Comic	9
CelebA HQ	9
<b>Vorkenntnisse und Arbeitsteilung</b>	<b>10</b>
<b>Fazit und Ausblick</b>	<b>10</b>
<b>Quellenverzeichnis</b>	<b>12</b>

# Einleitung

Künstliche Intelligenz ist heutzutage allgegenwärtig. Sowohl in selbstfahrenden Autos, als auch für die Sprachsteuerung unserer Elektrogeräte brauchen wir künstliche Intelligenz. Genauer gesagt Machine Learning. Ein besonders interessantes Teilgebiet der künstlichen Intelligenz ist das generative Deep Learning. Das generative Deep Learning ermöglicht es einem Computer unter anderem Bilder zu erzeugen, die von der Realität nicht zu unterscheiden sind, mitunter sogar Bilder von Menschen. Das ist nicht nur interessant, weil ein Computer das Gesicht eines Menschen kreiert, der so nie existiert hat, sondern ist auch von wirtschaftlicher Bedeutung. Durch künstlich generierte Bilder können teure, reale Gesichtsbilder umgangen, und Urheberrechtsverletzungen verhindert werden.

Das Ziel unserer Arbeit ist also das Erarbeiten einer künstlichen Intelligenz, die dazu in der Lage ist, menschliche Gesichter zu generieren. Als Trainingsdaten für das neuronale Netz verwenden wir zunächst einen Datensatz, welcher Fotos von menschlichen Gesichtern enthält. Das neuronale Netz soll nach erfolgreichem Training eine Bilddatei erzeugen, die von den Bildern, die im Datensatz enthalten sind, nicht zu unterscheiden ist. Das generierte Bild soll hierbei nicht nur eine Kopie oder Abwandlung eines bestehenden Bildes sein, sondern ein komplett neues und eigenes Gesicht darstellen.

# Datensätze

Da wir zunächst eine künstliche Intelligenz implementieren wollen, die Gesichter von Menschen generiert, verwenden wir CelebA [1] als Datensatz. Der Datensatz besteht aus Bildern, welche aus dem Internet entnommen wurden. Das Team des MMLAB der chinesischen Universität Hongkong das den Datensatz zusammengestellt, welcher für nicht kommerzielle Zwecke frei verfügbar ist. Der Datensatz beinhaltet über 200.000 Bilder von Prominenten. Die Bilder stammen von 10.177 verschiedenen Prominenten und beinhalten jeweils fünf Orientierungspunkte. Diese helfen dabei, eine einheitliche Ausrichtung der Bilder zu gewährleisten. Die ursprüngliche Auflösung der Bilder beträgt 178 x 218 Pixel. Hierbei war in unserem Fall eine Verarbeitung notwendig, bei der die Bilder so zugeschnitten wurden, dass eine Auflösung von 128x128 Pixel erreicht wurde. Hierbei ist anzumerken, dass die Bilder quadratisch sind und die Pixelzahlen Zweierpotenzen sind. Pro Bild gibt es 40 Attribute, welche die Person auf dem Bild beschreiben. Mit diesen Attributen lassen sich

weitere spannende Arbeiten gestalten. So könnte statt einem zufälligen Gesicht ein Gesicht generiert werden, welches bestimmte Merkmale enthält (z.B. Mann mit Bart und Brille). Zudem gibt es noch einen CelebA Datensatz, mit hochauflösenden Bildern. Wir konzentrieren uns bei der Implementierung unseres Modells zunächst auf den standard CelebA Datensatz. Das Modell haben wir jedoch neben CelebA mit weiteren Datensätzen trainiert, die Datensätze und die generierten Bilder werden in den Ergebnissen erläutert und präsentiert.

Zu den weiteren Datensätzen gehört ein Anime Datensatz. Dieser Datensatz beinhaltet 21551 Bilder, welche jeweils ein Gesicht im Anime-Stil abbilden [2]. Auch hier möchten wir eine neue Figur erschaffen, die den Figuren aus dem Datensatz ähnelt, jedoch keine Kopie einer bestehenden Figur ist. Zudem haben wir unser Modell mit einem Simpsons Datensatz [3] trainiert. Dieser Datensatz beinhaltet 9877 Bilder der Simpsons Staffeln 25 bis 28. Das

Ziel hierbei wäre es, eine neue Figur zu kreieren, welche Merkmale der Figuren aus dem Simpsons-Universum aufweist, jedoch nicht eine Kopie einer bestehenden Simpsons-Figur ist. Ein weiterer Datensatz, mit dem wir unser Modell trainiert haben, ist der Comic Datensatz [4]. Dieser Datensatz besteht aus 10.000 Bildern im

Comic-Stil. Das Ziel ist hierbei das selbe, wie bei den bisherigen Datensätzen auch. Der letzte Datensatz, mit dem wir unser Modell trainiert haben, ist der CelebA High Quality Datensatz [5]. Dieser besteht aus 30000 Bildern von Prominenten, in einer relativ hohen Auflösung (1024x1024). Es sind jeweils fünf Beispielbilder aus den jeweiligen Datensätzen in Abbildung 1 abgebildet.

## Architektur und Implementierung

Das GAN (Generative Adversarial Network) gehört mit den Auto Encodern momentan zu den am weitesten verbreiteten generativen Modellen. Es gibt mehrere Arten von GANs die jeweils andere Anwendungsgebiete abdecken. Die bekanntesten sind hierbei das Style GAN und das DC-GAN. Für unsere Arbeit haben wir das DC-GAN als Modell verwendet,

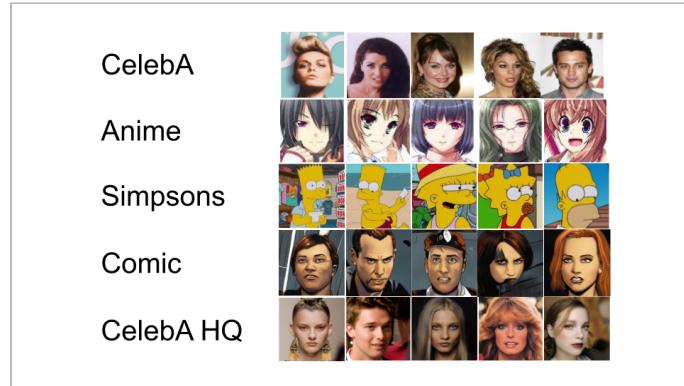


Abbildung 1: Beispiele aus den verwendeten Datensätzen.

welches im Vergleich zum Style GAN eher für Bilder mit einer niedrigeren Auflösung ausgelegt ist (Aila et al., 2019) [6]. In den folgenden Abschnitten erläutern wir zunächst die Funktionsweise eines gewöhnlichen GANs, um das Konzept zu verdeutlichen. Daraufhin kommen wir auf das DC-GAN zu sprechen und abschließend vergleichen wir die beiden Modelle.

## GAN

Ein GAN besteht zunächst aus 2 Modellen, die gegeneinander antreten. Hierbei gibt es den Generator, bei dem die Analogie zu einem "Künstler" gezogen werden kann. Der Generator wird trainiert um Bilder zu generieren. Die zweite Komponente hingegen, der Diskriminatator, kann als "Kunstkritiker" dargestellt werden. Der Diskriminatator wird trainiert um reale Bilder von generierten Bildern möglichst genau zu differenzieren (Goodfellow et al., 2014) [7]. Die angesprochenen Trainingsphasen laufen bei den beiden Komponenten jedoch sehr verschieden ab. Mit einer steigenden Epochen Anzahl lernt der Generator immer besser "reale" Bilder zu erzeugen. Der Diskriminatator hingegen lernt durch zunehmende Epochen Anzahl, reale Bilder von Fälschungen zu unterscheiden. Ein "Gleichgewicht" ist erreicht, sobald der Diskriminatator reale Bilder nicht mehr von Fälschungen unterscheiden kann. Um ein besseres Verständnis von GANs zu erlangen hilft es, sich die Eingaben und Ausgaben der jeweiligen Komponenten genauer anzuschauen. Der Generator bekommt als Eingabe einen Vektor von Zufallszahlen (random seed). Dieser seed wird vom Generator als Verarbeitungs Basis genutzt, um ein Bild zu generieren. Die einzige Aufgabe des Diskriminators ist es nun zu bewerten, mit welcher Wahrscheinlichkeit das Bild real ist. Dazu erhält der Diskriminatator ein Bild als input. Die Ausgabe des Diskriminators ist nun eine errechnete Wahrscheinlichkeit, ob das eingegebene Bild real ist [7].

Das Training der beiden künstlichen neuronalen Netze findet getrennt statt. Würde der Diskriminatator nur generierte Bilder bekommen, würde er schnell merken, dass er jedes Bild als Fälschung bewerten kann und somit eine Genauigkeit von 100% erzielt. Hier kommt der Trainingsdatensatz ins Spiel. Anstatt den Diskriminatator nur generierte Bilder bewerten zu lassen, kommen nun auch reale Bilder hinzu. So kann der Diskriminatator lernen, welche Eigenschaften ein reales Bild besitzt. Das führt dazu, dass der Diskriminatator die generierten Bilder des Generators erkennt. Ein präziser Diskriminatator ermöglicht es dem Generator, besser zu werden. Das Prinzip eines Trainings Durchlaufs ist in Abbildung 2 zu sehen.

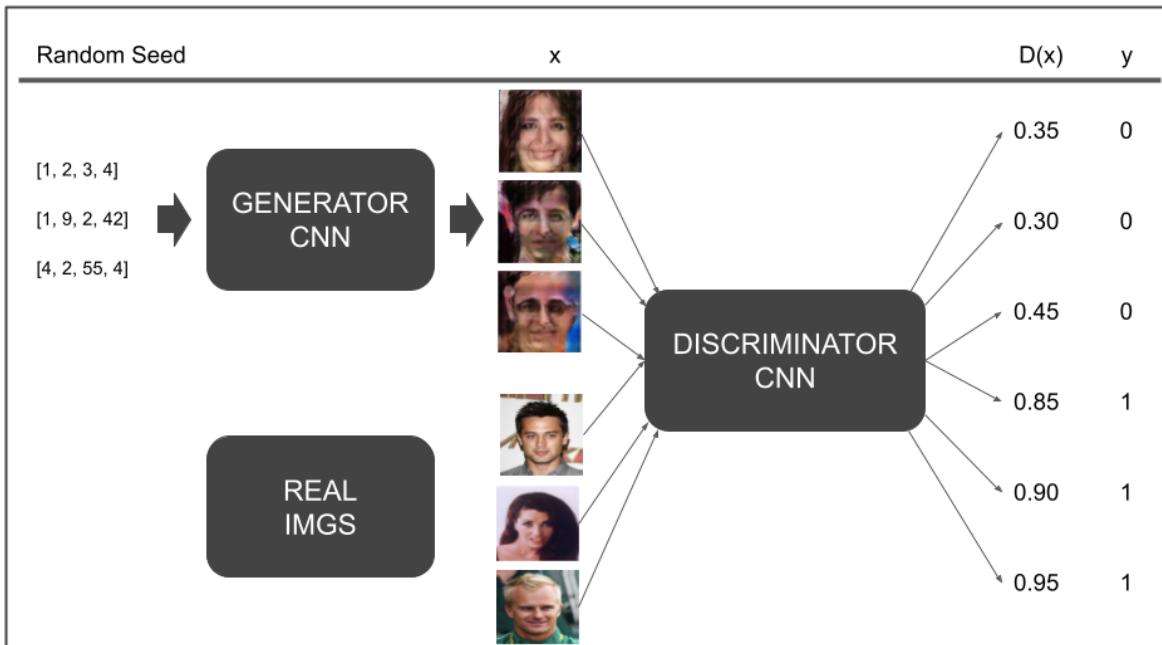


Abbildung 2: Ausgegebene Werte nach jeder Epoche.

## DCGAN

Das DC-GAN ist eine Erweiterung des Grundmodells GAN (Chintala et al., 2016) [8]. Beim DC-GAN werden im Generator convolutional Layers verwendet, um mit dem Eingabevektor das Ausgabebild zu generieren. Zusätzlich werden im Diskriminatator deconvolutional layers verwendet, um das Eingabebild besser klassifizieren zu können. Um präzise Ergebnisse zu erzielen, müssen die Gefalteten Neuronalen Netze nach einen bestimmten Architektur aufgebaut werden. Die Arbeitsgruppe aus der besagten wissenschaftlichen Publikation ist hierbei wie folgt vorgegangen:

Zunächst wurden Generator und Diskriminatator mit einer anderen Downsampling Strategie ausgestattet. Hierbei wurden die Forschungsergebnisse von all convolutional net verwendet, welche dem Netzwerk eine Optimierung des eigenen spatial downsamplings ermöglichen (Brox et al., 2015) [9]. Des Weiteren wurden vollständig miteinander verbunden Layer (fully connected layers) reduziert. Die ersten Schichten des Modells sind zwar vollständig miteinander verbunden, jedoch wird die vollständige Verbundenheit nicht in den weiteren Schichten fortgeführt. Das führt zu einer besseren Konvergenzgeschwindigkeit (convergence speed), schadet jedoch der Modell Stabilität. Die dritte Technik, die angewandt wurde, ist die Batch Normalization (Ioffe & Szegedy, 2015) [10]. Durch die Normalisierung der eingabe kann ein Kollabieren des Generators hinausgezögert oder ganz verhindert werden.

## GAN vs. DCGAN

Im Vergleich weisen die beiden Methoden jeweils Vor- und Nachteile auf. Das gewöhnliche GAN ist relativ generisch und Charakteristiken für die Modellspezifikationen werden offen gelassen. Das standard GAN ist Attraktiv für Repräsentationslernen und kann besser auf eine breitere Domäne angewendet werden [8]. Jedoch gelten GANs als instabil, womit Generatoren oft sinnfreie Outputs erzeugen.

Bei DC-GANs hingegen steht die Fokussierung auf räumliche Korrelationen im Vordergrund. Somit besitzt das Modell ein weniger breites Anwendungsgebiet, kann jedoch besser für Bilder und Videos verwendet werden. Jedoch hat auch das DC-GAN Limitation. So kann es durch lange Trainingszeiten dazu kommen, dass das Modell eine Teilmenge von Filtern zu einem einzigen Filter kollabiert, was einem oszillierenden Modus ähnelt [8]. Für unseren Anwendungsfall ist also das DC-GAN besser geeignet als das Standardmodell.

## Implementierung und Parameter

Unser Vorgehen war es zunächst eine DC-GAN Architektur zu implementieren und mit dem CelebA Datensatz zu trainieren. Hierzu gibt es eine hohe Anzahl an Quellen und viele Implementationen mit voreingestellten Parametern.

Allgemein	Generator	Diskriminator	Adam Optimizer
Momentum = 0.9	Lernrate: 0.0002	Lernrate = 0.0002	$\beta_1 = 0.5$
Epsilon = 0.0005	Filter: [32, 64, 128, 256, 512]	Filter: [1024, 512, 256, 128, 64]	$\beta_2 = 0.999$

Abbildung 3: Parameter für die Architektur.

Unsere Implementation und Parameterauswahl basieren auf dem "DCGAN-TF2.0" Github Repository (Bingi, 2021) [11], welches wiederum auf mehreren

wissenschaftlichen Publikationen [12, 13, 14] basiert. Für unser Modell haben wir Python 3.6 und das TensorFlow 2.5 Framework in Kombination mit CUDA 11.3 verwendet. Nachdem wir die Implementierung der Architektur an unsere Hardware angepasst und trainiert haben, bekamen wir erstmals solide Ergebnisse. Nach einer Optimierungsphase, in der verschiedene Parameterkombinationen ausprobiert wurden, haben wir einen Parametersatz festgelegt. Davon sind die wichtigsten Parameter in Abbildung 3 zu sehen.

Das Epsilon wurde mit 0.0005 festgelegt und dient dazu, eine Division durch Null zu vermeiden. Die Lernraten für Generator und Diskriminator betragen jeweils 0.0002. Eine höhere Lernrate würde das Risiko erhöhen, dass das Modell kollabiert. Die Adam Optimizer haben die Werte  $\beta_1 = 0.5$  und  $\beta_2 = 0.999$ . Diese Werte werden benötigt für den Adam Lernraten-Optimierungsalgorithmus. Für gewöhnlich setzt man die Hyperparameter auf hohe

Werte ( $> 0.5$ ). Die Filter für die convolutional layers im Generator sind aufsteigende Zweierpotenzen von 32 bis 512, während die deconvolutional layers im Discriminator absteigende Zweierpotenzen von 1024 bis 32 zugewiesen bekommen haben.

Abschließend haben wir weitere Datensätze ausgewählt, welche verschiedene Bilder und Auflösungen enthalten. Das Modell wurde mit jedem Datensatz für 100 Epochen trainiert. Nach jeder Epoche haben wir Zwischenergebnisse und gemessene Zeiten ausgeben lassen. Anschließend wurden die Ergebnisse evaluiert.

## Ergebnisse

Wie bereits erwähnt haben wir unser Modell mit jedem Datensatz 100 Epochen lang trainiert. Während den Trainingsdurchläufen haben wir für jede Epoche generierte Bilder, gemessene Zeiten und Lossfunktionswerte ausgegeben. Die Lossfunktionswerte haben wir

für jeden Datensatz in einem Graphen zusammengefasst. Die Trainingsdauer, die Batches und den Speicherbedarf der serialisierten Datensätze haben wir in einer Tabelle zusammengetragen, welche in Abbildung 4 zu sehen ist. Im

	CelebA	Anime	Simpsons	Comics	CelebAHQ
Dauer	18.61 h	0.98 h	1.31 h	9.51 h	<b>71.83 h</b>
Batches	<b>632.760</b>	67.300	30.800	31.200	187.500 <sup>2</sup>
Größe <sup>1</sup>	9.28 GB	0.25 GB	1.80 GB	7.32 GB	<b>87.80 GB</b>

1: Größe der serialisierten Datensätze (TF Records)

2: Batchgröße von 32 auf 16 reduziert wegen Auflösung

Abbildung 4: Ausgegebene Werte nach jeder Epoche.

Folgenden werden wir die Ergebnisse für jeden Datensatz evaluieren und gegebenenfalls den Trainingsverlauf anhand der Lossfunktionswerte erläutern.

## CelebA

Abbildung 5 zeigt die ausgegebenen Bilder für einige ausgewählte Epochen aus dem

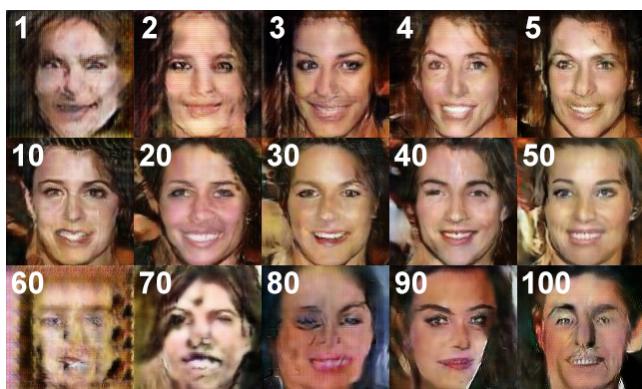


Abbildung 5: Bilder aus ausgewählten Epochen.

Trainingsverlauf. Das Modell generiert zunächst zufällige Pixel, die einem menschlichen Gesicht nicht ähnlich sehen, wird jedoch rasant besser. Zunächst verwischt das Gesicht nicht mehr mit dem Hintergrund und klare Konturen sind zu erkennen. Von Epoche 20 bis 50 findet eine Konstante Verbesserung der Ergebnisse statt,

welche kurz nach Epoche 50 ihr Optimum erreicht zu haben scheint. Hier sehen die Bilder

sehr real aus, wie man in Abbildung 6 mit den Ausgaben nach Epoche 50 sehen kann, und sind teilweise nicht mehr von echten Menschen zu unterscheiden.



Abbildung 6: Real ausschende CelebA Trainingsergebnisse.

Nach Epoche 50 werden die Bilder plötzlich viel unrealistischer. Im weiteren Verlauf verbessern Sie sich etwas, jedoch werden mehrmals Bilder generiert, die sich sehr ähnlich sehen. Das Phänomen heißt Mode Collapse und tritt häufig bei GAN Architekturen auf (Google LLC, 2021) [15]. Abbildung

7 zeigt die Lossfunktionswerte unseres Modells, mit denen sich das Phänomen anschaulich erklären lässt. Auf dem Graphen sind die jeweiligen Lossfunktionswerte für den Generator (rot) und den Diskriminatator (blau) über 100 Epochen angegeben. Hierbei möchte der Generator seinen Lossfunktionswert maximieren, während der Diskriminatator ein gutes Ergebnis erzielt hat, sobald sein Lossfunktionswert niedrig ist. Der Lossfunktionswert bewertet also die Performance der einzelnen Komponenten. Kurz vor der 60. Epoche sieht man einen rasanten Anstieg des Generators. Somit war der Generator in dieser Epoche um einiges besser als der Diskriminatator und konnte ihn somit "ausspielen". Da der Generator hier scheinbar ein sehr gutes Bild generiert hat, werden bestimmte Werte festgesetzt und in Zukunft bei allen generierten Bildern angewandt. Die Bilder sehen jedoch nicht wirklich real aus, sie wurden nur vom Diskriminatator als real bewertet. Das führt dazu, dass die in Zukunft generierten Bilder schlechtere Ergebnisse aufweisen.

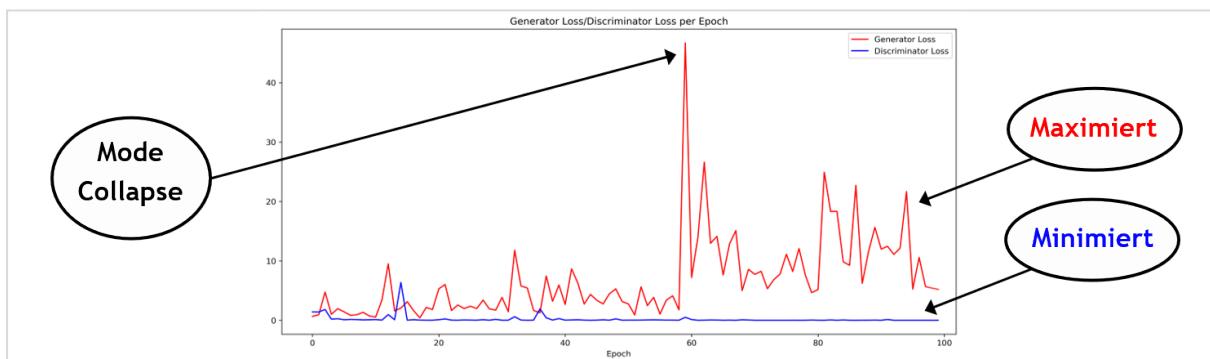


Abbildung 7: Loss-Funktionswerte nach jeder Epoche.

## Anime



Abbildung 8: Generierte Bilder nach Training mit dem Anime Datensatz.

Während das Modell mit dem Anime Datensatz trainiert wurde, hat es sich zu Beginn relativ schnell verbessert. Das Optimum war jedoch nach einer kurzen Epochenzahl erreicht, danach blieb die Qualität der generierten Bilder konstant. Die Ergebnisse waren gut und die

generierten Bilder waren kaum zu unterscheiden von den Bildern aus dem Datensatz. Einige Bilder des Trainings von Epoche 100 sind in Abbildung 8 zu sehen. Lediglich kleine Artefakte hinsichtlich einzelner Pixel sind zu erkennen. Während den späteren Epochen sehen sich einige Bilder ähnlich, was auf einen Mode Collapse hindeutet.

## Simpsons

Unsere Architektur hat während dem Training mit den Simpsons Daten weniger gute

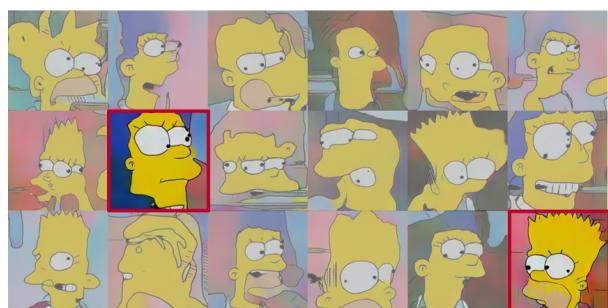


Abbildung 9: Generierte Bilder nach Training mit dem Simpsons Datensatz.

Ergebnisse erzielt. Der Generator hat meistens Bilder generiert, die nicht wie ein Simpsons Charakter ausgesehen haben. In den seltenen Fällen, in denen der Generator ein Bild generiert hat, welches aus dem Simpsons Universum stammen könnte, war es nur eine Kombination

zwischen zwei, bereits existierenden, Simpsons Charakteren (zum Beispiel Homer und Bart). Daher kamen wir zu der Erkenntnis, dass der Simpsons Datensatz nicht gut geeignet für unser Modell ist. Es sind zwar genug Bilder in ausreichender Qualität vorhanden, die geringe Anzahl an verschiedenen Simpsons Charakteren limitiert die Ergebnisse jedoch. In Abbildung 9 sind Ausschnitte aus dem Trainingsergebnis nach Epoche 100 mit dem Simpsons Datensatz zu sehen. Eine weitere Ursache für die mittelmäßigen Ergebnisse kann auch die große Varianz zwischen den einzelnen Gesichtern sein (der Unterschied zwischen Homer und Marge ist extremer als bei echten Menschen).

## Comic

Da die Bilder des Comic Datensatzes in einer relativ hohen Auflösung sind, haben wir weniger gute Ergebnisse erwartet. Jedoch waren die Ergebnisse des Comic Datensatzes die



Abbildung 10: Generierte Bilder nach Training mit dem Comic Datensatz.

besten Ergebnisse, die wir im Rahmen dieser Projektarbeit erzielen konnten. Die generierten Bilder haben sich konstant verbessert und waren irgendwann nicht mehr von den Bildern aus dem Comic Datensatz zu unterscheiden. In Abbildung 10 sind die Ergebnisse von Epoche 100 zu

sehen. Einige kleine Auffälligkeiten sind Artefakte bei Haaren, bei manchen Bildern sind aber auch selbst die ohne Probleme. Die guten Ergebnisse sind vermutlich auf den geeigneten Datensatz zurückzuführen, welcher aus vielen ähnlichen Gesichtern besteht, die keine starke Varianz wie bei den Simpsons aufweisen.

## CelebA HQ

Der CelebA High Quality Datensatz hat eine Auflösung von 1024x1024 Pixeln. Somit haben



Abbildung 11: Generierte Bilder nach Training mit dem CelebA HQ Datensatz.

die Bilder dieses Datensatzes die größte Auflösung, mit der wir unser Modell trainiert haben. Am Anfang des Trainingsverlauf ist eine Verbesserung zu erkennen, jedoch stagniert der Fortschritt nach ca. 20 Epochen. Die Bilder nach Epoche 100 sind in Abbildung 11 abgebildet und lassen sich

schnell als generiert identifizieren. Die schlechten Ergebnisse sind der hohen Auflösung der Bilder geschuldet. DC-GANs sind nicht auf eine so hohe Auflösung ausgelegt. Die Trainingsdauer war enorm und die serialisierten Datensätze haben fast 90 GB Speicher benötigt. Um Bilder mit einer hohen Auflösung zu generieren, eignen sich andere Architekturen wie das Style GAN besser.

# Vorkenntnisse und Arbeitsteilung

## Vorkenntnisse

Name	Studiengang	Erfahrung
Christian Loor	M.Sc. Management	Programmierkenntnisse
Tomislav Pree	M.Sc. Wi. Informatik	Programmierkenntnisse
Kevin Kammler	M.Sc. Wi. Informatik	Programmierkenntnisse, Machine Learning, Einführung in die künstl. Intelligenz

Zu Beginn der Arbeitszeit haben wir uns zunächst in die Thematik eingeleSEN und wöchentlich abgesprochen. Nach anfänglicher Informationsbeschaffung haben wir gemeinsam die Ziele unserer Projektarbeit festgelegt. Die Implementierung erfolgte zunächst bei jedem Gruppenmitglied lokal. Ein regelmäßiger Abgleich garantierte hierbei, dass wir nicht aneinander vorbei arbeiten. Nachdem jeder verschiedene Modelle lokal implementiert und trainiert hatte, haben wir uns auf ein Modell geeinigt und die Hardware bestimmt, auf der die Architektur trainiert werden sollte. Nachdem unser Grundmodell stand und mit dem CelebA Datensatz trainiert wurde, hat Kevin sich darauf konzentriert, das Modell mit den weiteren Datensätzen zu trainieren. Währenddessen hat Christian hauptsächlich die Präsentation konzeptioniert und Tomislav hat sich auf die Dokumentation konzentriert. Die einzelnen Arbeitsschritte sind jedoch immer mit der gesamten Gruppe besprochen worden.

## Fazit und Ausblick

Das Modell generiert relativ schnell gute Ergebnisse bei niedrigen bis mittleren Auflösungen (CelebA, Anime, Comics) und ist auf verschiedene Arten von Gesichtern anwendbar (real und gezeichnet). Jedoch ist die passende Wahl des Datensatzes wichtig für gute Ergebnisse, ein ungeeigneter Datensatz (Simpsons) kann daher schnell zu schlechten

Bildern führen. Eine zu hohe Epochenanzahl führt auch beim DC-GAN zum Kollabieren und das Training für hochauflösende Bilder (CelebA HQ) dauert sehr lange.

Es gibt jedoch Methoden, und Erweiterungen die man anwenden kann, um die bisherigen Mängel auszugleichen. So kann das Kollabieren der Architektur mit alternativen GANs wie den Unrolled GANs oder alternativen Loss-Funktionen wie dem Wasserstein-Loss umgangen werden. Die hohen Trainingszeiten mit dennoch schlechten Resultaten für hochauflösende Bilder lassen sich umgehen, indem man auf das StyleGAN zurückgreift, welches besser für hochauflösende Bilder geeignet ist.

Interessante Erweiterungen der Projektarbeit wären zunächst die Berücksichtigung von Bildattributen. Hierbei könnte der Nutzer Parameter bestimmen, welche das generierte Bild enthalten soll. Das Modell generiert dann ein entsprechendes Bild, beispielsweise einen Mann mit Brille und Bart, wobei dafür ein Datensatz wie CelebA mit Labels vorausgesetzt werden würde. Die Verwendung von anderen Trainings Datensätzen stellt ebenfalls eine mögliche Erweiterung dar. So könnte man auch nicht-menschliche Gesichter, zum Beispiel Tiergesichter, oder generell Objekte wie Häuser generieren. Sobald ein passender Datensatz vorliegt, sind der Fantasie hierbei keine Grenzen gesetzt.

# Quellenverzeichnis

[1] Multimedia Laboratory, The Chinese University of Hong Kong. (2015).

*Large-scale CelebFaces Attributes (CelebA) Dataset* [Datensatz].

<https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html>

[2] *Anime Faces*. (2019, 16. Mai). [Datensatz].

<https://www.kaggle.com/soumikrakshit/anime-faces>

[3] *Simpsons Faces*. (2018, 28. September). [Datensatz].

<https://www.kaggle.com/kostastokis/simpsons-faces>

[4] *Comic faces (paired, synthetic) v2*. (2021, 14. Mai). [Datensatz].

<https://www.kaggle.com/defileroff/comic-faces-paired-synthetic-v2>

[5] *celeb\_a\_hq | TensorFlow Datasets*. (2018). [Datensatz].

[https://www.tensorflow.org/datasets/catalog/celeb\\_a\\_hq](https://www.tensorflow.org/datasets/catalog/celeb_a_hq)

[6] Aila, A., Laine, S., Karras, T. (2019). *A Style-Based Generator Architecture for Generative Adversarial Networks*. CVPR, Long Beach, Kalifornien

[7] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y. (2014). *Generative Adversarial Nets*. In Advances in Neural Information Processing systems (pp. 2672–2680).

[8] Chintala, S., Metz, L. & Radford, A. (2016). *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. ICLR, San Juan, Puerto Rico.Long Beach

[9] Brox, T., Dosovitskiy, A., Riedmiller, M., Springenberg, J. (2015). *Striving for Simplicity: The All Convolutional Net*. ICLR, Freiburg, Deutschland.

[10] Sergey Ioffe, & Christian Szegedy. (2015). *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*.

[11] Bingi, A. *DCGAN-TF2.0*. Stand: 01.07.2021. Verfügbar unter:

<https://github.com/adityabingi/DCGAN-TF2.0>

[12] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, & Yoshua Bengio. (2014). *Generative Adversarial Networks*.

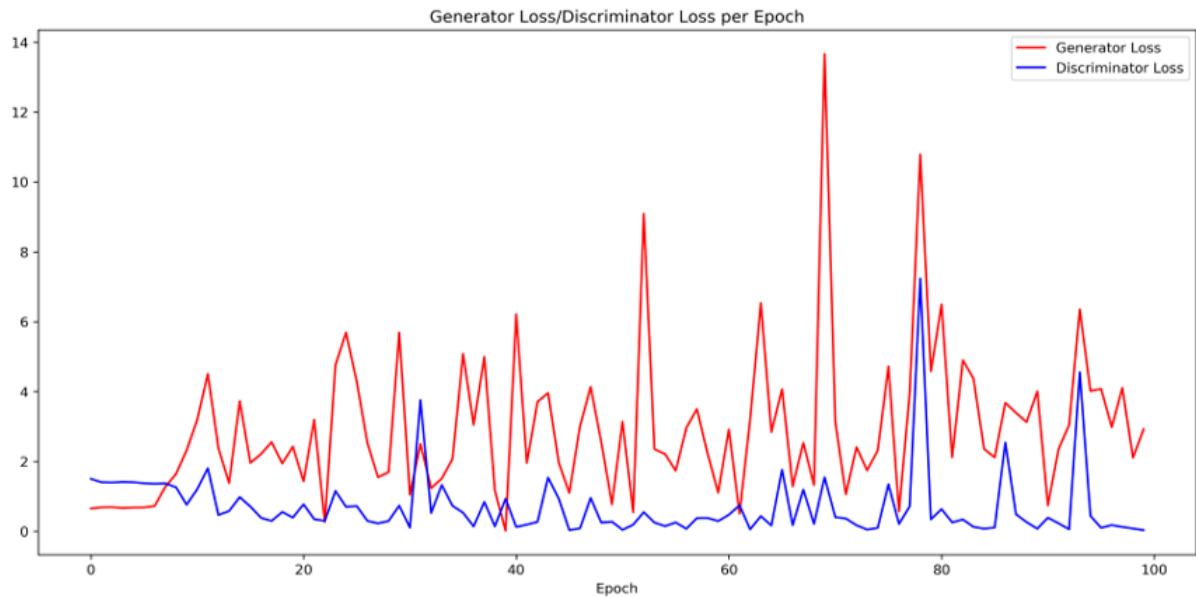
[13] Ian Goodfellow. (2017). *NIPS 2016 Tutorial: Generative Adversarial Networks*.

[14] Alec Radford, Luke Metz, & Soumith Chintala. (2016). *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*.

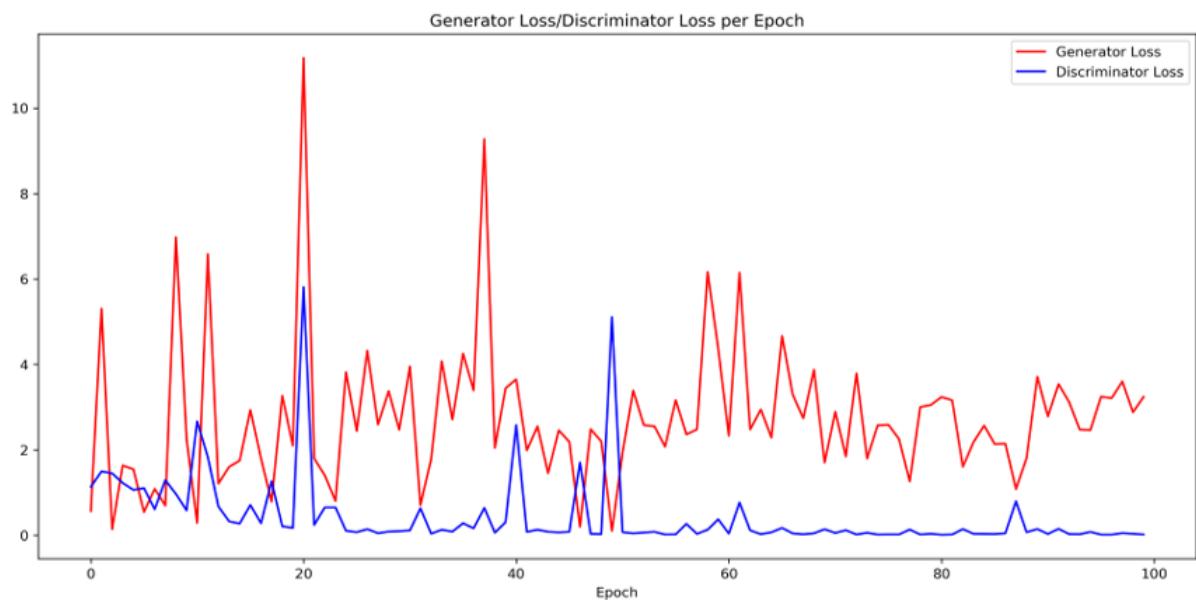
[15] Google LLC. *Real World GANs – Common Problems*. Stand: 01.07.2021. Verfügbar unter:  
<https://developers.google.com/machine-learning/gan/problems>

# Anhang

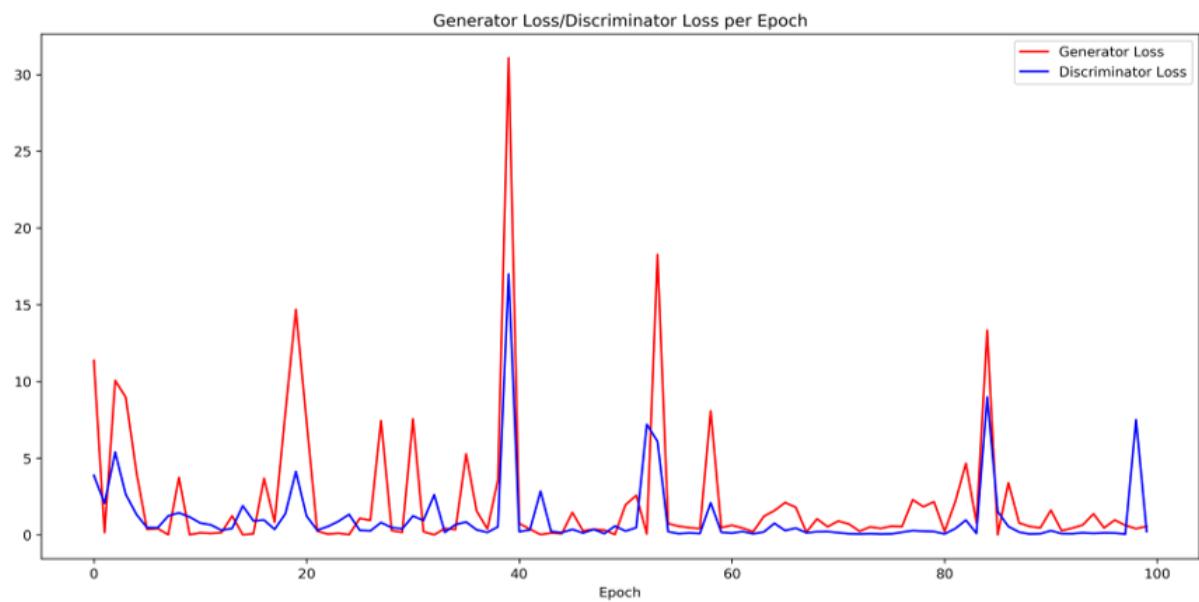
Anhang 1: Werte der Lossfunktion für den Anime Datensatz.



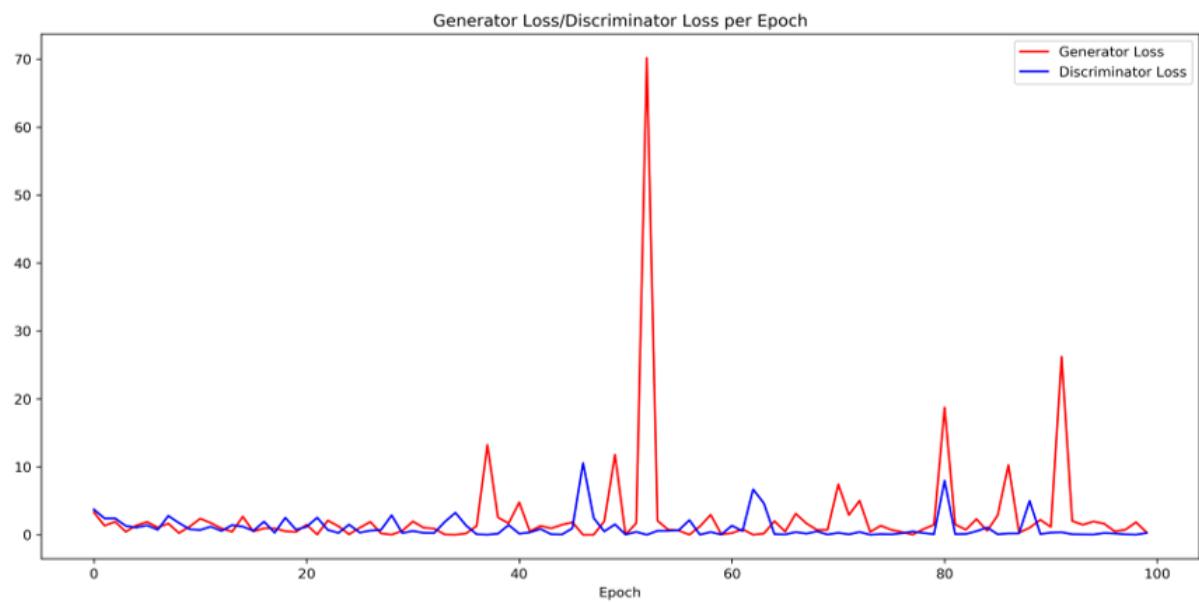
Anhang 2: Werte der Lossfunktion für den Simpsons Datensatz.



### Anhang 3: Werte der Lossfunktion für den Comic Datensatz.



### Anhang 4: Werte der Lossfunktion für den CelebA HQ Datensatz.



# Eidesstattliche Erklärung

„Ich versichere, dass ich die Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen oder anderen Quellen (auch Internet) entnommen sind, habe ich als solche eindeutig kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht und noch nicht als Studienleistung zur Anerkennung oder Bewertung vorgelegt worden. Mir ist bekannt, dass Verstöße gegen diese Anforderungen zur Bewertung der Arbeit mit der Note „Nicht ausreichend“ führen sowie die Nichterteilung des angestrebten Leistungsnachweises zur Folge haben.“

16.07.2021

---

Datum



---

Christian Loor



---

Kevin Kammler



---

Tomislav Pree