Evaluierung des Lernerfolgs eines GANs mittels Generierung von geometrischen Figuren

Studienarbeit

des Studienganges Informatik

an der Duale Hochschule Baden-Württemberg Karlsruhe

von

Jonas Bürgel und Patrick Welter

Abgabedatum:	15. Mai 2022
Kurs:	TINF19B4
Betreuer:	Markus Reischl

Eidesstattliche Erklärung

Erklärung gemäß § 5 (3) der 'Studien- und Prüfungsordnung DHBW Technik' vom 29. September 2015. Ich versichere hiermit, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen als Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Karlsruhe, den 7. März 2022

Jonas Bürgel und Patrick Welter

Sperrvermerk

Der Inhalt dieser Arbeit darf weder als Ganzes noch in Auszügen Personen außerhalb des Prüfungsprozesses und des Evaluationsverfahrens zugänglich gemacht werden, sofern keine anders lautende Genehmigung der Ausbildungsstätte vorliegt.

Copyright-Vermerk

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

2 Stand der Technik

2.1 General Adversarial Network

Der Begriff GAN *(General Adversarial Network)* ist auf Ian Goodfellow zurückzuführen [1]. Das Wort bezeichnet ein Konstrukt aus 2 neuronalen Netzen, die sich gegenseitig trainieren. Durch das spezielle Training gelingt die Generierung von realistischen unechten Daten. Solche Daten können dann zum Beispiel für das Training anderer neuronalen Netze [2], in der Bildverarbeitung [3, 4] und vielen weiteren Anwendungsgebieten verwendet werden [5, 6].

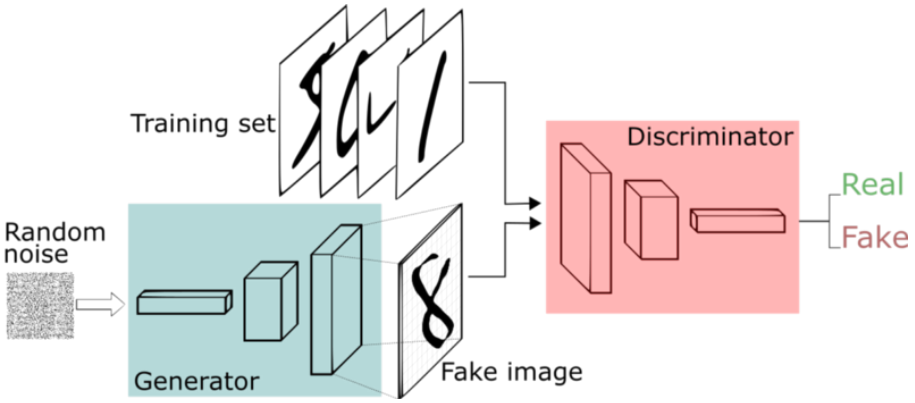

Abbildung 2.1: Generative Adversarial Network (Bild von Thalles Silva [7])

Die beiden Netze eines GANs werden in den Generator und den Discriminator unterschieden. Aufgabe des Generators ist die Generierung von unechten Daten. Dafür wandelt er eine zufälligen Eingabe in einen möglichst realistischen Output um. Die zufällige Eingabe dient dabei als Basis für die Ausgabedaten. Das ist notwendig, da der Umwandlungsprozess selbst deterministisch ist, aber trotzdem eine Vielzahl an unterschiedlichen Daten generiert werden soll.

Der Output des Generators wird vom Discriminator klassifiziert. Dafür wird er sowohl auf die generierten Daten als auch einen Bestand an echten Daten trainiert. Sein Ziel ist es dann, die falschen Daten des Generators zu identifizieren. Ziel des Generators hingegen ist es, den Discriminator zu täuschen und die generierten Daten als echt wirken zu lassen.

Ian Goodfellow bezeichnet den Lernprozess auch als Minimax-Spiel, bei die Ausgabe des Discriminators die zu optimierende Größe ist. Das bedeutet, der Generator versucht die Genauigkeit des Discriminators zu verringern, während der Discriminator sie erhöhen möchte. [8]

Für diese Arbeit werden zusätzlich Label eingeführt, um die generierten Daten beeinflussen zu können. Diese Art von GAN nennt sich CGAN oder Conditional GAN [9]. Dazu erhalten der Generator und Discriminator das Datenlabel als einen weiteren Input. So kann der Discriminator aus den echten Daten lernen, dass Daten bei bestimmten Labeln bestimmte Eigenschaften haben. Dadurch ist dann der Generator gezwungen, diese Eigenschaften zu berücksichtigen, um den Discriminator wieder zu täuschen.

2.2 Verfahren zur Bestimmung von Hyperparametern

Quelle [10]

Hyperparameter dienen der Konfiguration des Trainingsprozesses. Im Gegensatz zu den Parametern neuronaler Netze werden sie durch das Training nicht verändert. Bei Hyperparametern handelt es sich zum Beispiel um die Lernrate oder Lossfunction [11]. Die Bestimmung von Hyperparametern ist wichtig für den Trainingserfolg des Modells und sollte für jeden Datensatz neu bestimmt werden.

Jedoch sind nicht alle Hyperparameter gleichbedeutend für den Trainingserfolg, beispielsweise die Lernrate zählt zu den dominanteren Faktoren [12]. Zusätzlich unterscheiden sich optimale Hyperparameter bei verschiedenen Datensätzen oder Netzarchitekturen. Für die Suche nach optimalen Hyperparametern gibt es verschieden Verfahren, die im Folgend vorgestellt werden.

Manuelle Suche

Zunächst ist es möglich die Hyperparameter manuell festzulegen. Das ist vor allem bei unwichtigen Hyperparametern sinnvoll, da dort bekannte Standardwerte gesetzt werden

können. Für eine gründliche Suche aller Hyperparameter ist die manuelle Suche allerdings viel zu aufwändig.

Gridsearch

Bei der Gridsearch handelt es sich um eine sehr traditionelle Technik und ist gut mit einem Brute-Force Ansatz vergleichbar [13]. Dafür werden zunächst die zu untersuchenden Hyperparameter ausgewählt. Danach erfolgt die Definition eines Suchintervalls inklusive Intervallschritten für jeden Hyperparameter. Schließlich wird das neuronale Netzwerk für jede Kombination der Hyperparameterwerte trainiert und die Ergebnisse der einzelnen Durchläufe festgehalten.

Nach dem Training ist es dann möglich, die einzelnen Ergebnisse zu vergleichen. Dabei können dann Schemata erkannt und Kombinationen aussortiert werden. Es ist dann möglich die erfolgversprechendsten Netze weiter zu trainieren, oder direkt eine zufriedenstellende Kombination auszuwählen.

Insgesamt ist Gridsearch sehr simpel zu implementieren, aber auch sehr ressourcenaufwändig. Der exakte Aufwand hängt dabei stark von der Anzahl der möglichen Kombinationen der Hyperparameterwerte ab. Weil es sich bei den Kombinationen um ein ein Kreuzprodukt handelt, wächst der Aufwand mit der Anzahl an Werten auch sehr stark an. Durch die Möglichkeit von Parallelisierung findet die Gridsearch in der Regel bessere Parameter als die sequentielle manuelle Suche in der gleichen Zeit [14].

Random Search

Die Random Search [14] funktioniert ähnlich wie die Gridsearch, nur werden zufällige Werte statt einem festgelegten Werteraster erzeugt (Abb. 2.2). Die zufälligen Werte haben den Vorteil, dass es weniger Werte-Überlagerungen im Vergleich zur den Raster-Kombinationen gibt. Deswegen kann die zufällige Suche bei gleich vielen Durchläufen ein größeres Spektrum an Ergebnissen abdecken. Mittels *Automatic Relevance Determination* [15] ist es dann möglich, den Einfluss und Wertebereich der einzelnen Hyperparameter darzustellen.

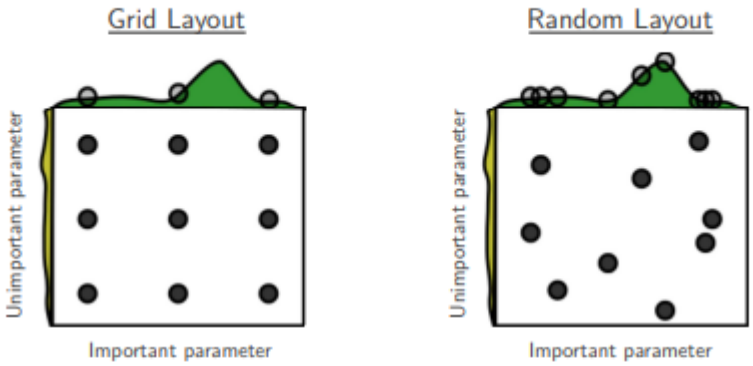

Figure 1: Grid and random search of nine trials for optimizing a function $f(x,y) = g(x) + h(y) \approx g(x)$ with low effective dimensionality. Above each square $g(x)$ is shown in green, and left of each square $h(y)$ is shown in yellow. With grid search, nine trials only test $g(x)$ in three distinct places. With random search, all nine trials explore distinct values of g. This failure of grid search is the rule rather than the exception in high dimensional hyper-parameter optimization.

Abbildung 2.2: Random Search vs Grid Search (Quelle [14])

Problem der Random Search ist vor allem, dass die Räume der guten Hyperparameter (*Search Space*) sehr klein sind. Dadurch ist nicht gewährleistet, dass ein zufälliger Wert auch den Raum trifft. Jedoch erzielen Optimierungen mittels Random Search in der Regel trotzdem (sehr) gute Ergebnisse. Dem Grid-Search-Verfahren ist die Random-Search allerdings nicht prinzipiell überlegen [14].

Genetic Algorithm - Evolutionäre Suche

mehr belegen

Das evolutionäre Suchverfahren wandelt die Hyperparametersuche in einen evolutinären Entwicklungsprozess um. Die 'Evolution' des Algorithmus ist eine Anlehnung an die natürliche Entwicklung in der Natur. Diese Entwicklung zeichnet sich durch die Operatoren Selektion, Kombination und Mutation aus, die auch der Algorithmus verwendet. [16, 17]

Für den Optimierungsprozess werden am Anfang mehrere zufällige Hyperparameterinstanzen erzeugt. Die Instanzen werden dann trainiert und stehen im Wettbewerb um das

beste Ergebnis. Nach der Auswertung der Trainingsergebnisse werden die besten Instanzen selektiert und miteinander gekreuzt und oder mutiert.

Die evolutionäre Suche eignet sich besonders gut bei sehr vielen Hyperparametern. Im Gegensatz zur Grid-Search werden nicht immer alle Kombinationen ausprobiert, sondern nur einzelne, die sich bereits zuvor als vielversprechend herausgestellt haben. Dadurch kann mit sehr großen Mengen an Hyperparametern umgegangen werden.

Allerdings ist die evolutionäre Suche vergleichsweise langsam, da die Evolutionsschritte sequentiell ablaufen müssen. Erst bei einer hohen Anzahl an Hyperparametern oder einem sehr großen Suchraum wird die Grid-Search dann so langsam, dass sich die evolutionäre Suche lohnt [16].

2.3 Related Work

Es gibt viele verschiedene Versuche und Varianten die Hyperparameter zu optimieren. Dementsprechend viele Papers und Artikel befassen sich damit. Einige dieser Arbeiten werden nachfolgend kurz vorgestellt.

Best Selection of Generative Adversarial Networks Hyper-Parameters Using Genetic Algorithm [17]

Die Studie befasst sich mit Hyperparameter Optimierung auf Basis von genetischen Algorithmen. Genetische Algorithmen zeichnen sich unter anderem durch die verwendeten Operatoren Mutation, Kreuzung und Selektion aus [18]. Ziel der Studie war es, bestmögliche Hyperparameter für das MNIST Datenset zu finden [19]. Dabei wurden die Hyperparameter *learning-rate, dropout, batch-size und number of neurons in dense layer* betrachtet.

Zwar sind die Daten und die Herangehensweise nicht die selbe wie in dieser Arbeit, aber die untersuchten Hyperparameter. Diese können mit als Basis für die Untersuchung genommen werden. Zudem verdeutlicht die Arbeit noch einmal die große Auswirkung von Hyperparametern.

Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks [20]

Ziel dieser Studie ist es allgemein gute Hyperparameter zu finden. Dafür wurden Bilder aus verschiedenen Datensets genommen (Large Scale Scene Understanding [21], Imagenet-1k [22] und einem eigenen Datenset mit Gesichtern). Bei den Hyperparametern wurden diverse untersucht, zum Beispiel aber nicht ausschließlich die Lernrate, Momentum und Batch-Size. Im Ergebnis konnten für alle Werte gefunden werden, die über alle Datensets meistens stabil mit zufriedenstellenden Ergebnissen liefen. Erkenntnis der Studie war auch, dass keine Hyperparameter Kombination gefunden werden konnte, die über alle Datensets 100% stabil laufen konnte.

Hyperpara learning-rate 0.0002, momen-tum (beta1) zu 0.5, mini-batch size 128,... (Kapi-tel 4 Details of adver-sarial trai-ning)

Eine allgemeine Studie zum Thema Hyperparameter ist für die Arbeit sehr interessant, da sie gute Ausgangswerte bietet. Dass keine 100% stabile Zusammenstellung gefunden werden konnte, verdeutlicht die Notwendigkeit der Feinanpassung an jedes Datenset.

The GAN Landscape: Losses, Architectures, Regularization, And Normalization [23]

Dieses Studie beschäftigt sich nicht ausschließlich mit Hyperparametertuning. Aber auch in dieser Studie werden allgemein GANs optimiert und die Erkenntnisse daraus können verwendet werden. In the GAN Landscape wird das GAN auf die Datensets CIFAR10 [24], CELEBA-HQ-128 und LSUN-BEDROOM [21]. Für das Training wurden dann ein DCGAN und ResNet verglichen. Dabei fand die Studie viele interssante Erkenntnisse heraus, zum Beispiel eine gute Updatequote von 5:1 zwischen Discriminator und Generator.

Besonders interessant bei dieser Arbeit ist die Bestimmung von anderen Methoden, wie das unterschiedliche Updaten von Discrimintaor und Generator. Wieder bietet die Arbeit eine gute Ausgangslage, aber keine Lösung für ein Datenset mit geometrischen Bildern.

Tunability: Importance of Hyperparameters of Machine Learning Algorithms

`https://www.jmlr.org/papers/volume20/18-444/18-444.pdf`

3 Methodik

In diesem Kapitel werden zunächst Eigenschaften der Trainingsdaten erläutert. Dabei wird auf die Bildeigenschaften und Besonderheiten der dargestellten Figuren in den Bildern eingegangen. Danach erfolgt eine Festlegung der Kriterien für ein erfolgreiche trainiertes 'GAN', das Grundlage für spätere Vergleiche von Trainingserfolgen sein wird.

3.1 Trainingsdaten

Bei den Trainingsdaten handelt es sich um synthetisch generierte Bilder mit geometrischen Figuren. Zwar gibt es bereits Datensätze mit solchen Bildern [25, 26], im Rahmen der Studienarbeit werden jedoch keine vorgefertigten Datensätze verwendet. Denn die Generierung eigener Bilder erlaubt eine größere Kontrolle über Eigenschaften der Bilder und Inhalte, als vorgefertigte Sets. Damit trotzdem eine Vergleichbarkeit zu anderen Arbeiten gewährleistet werden kann, wird die Generierung deterministisch reproduzierbar und dokumentiert sein.

3.1.1 Bildeigenschaften

Die Bildeigenschaften sind ein wichtiger Bestandteil des Datensatzes. Mit Bildeigenschaften sind dabei nicht Inhalte, sondern allgemeine Merkmale, wie Größe oder Farbe gemeint. Durch die Merkmale werden vor allem die benötigten Rechenkapazitäten zum Trainieren des GANs beeinflusst. So können Trainingszeiten durch kleinere Bilder verkürzt werden, da mehr Bilder auf einmal in den Grafikspeicher der GPU geladen werden können. Als Folge können mehr Konfigurationen ausprobiert werden Die Qualität der Bilder muss allerdings immer hoch genug bleiben, dass eine Erkennbarkeit der Formen gewährleisten ist.

Farbe

Die Bilder sind grau-skaliert, das heißt ein Pixel entspricht einer Zahl zwischen 0 (schwarz) und 255 (weiß).

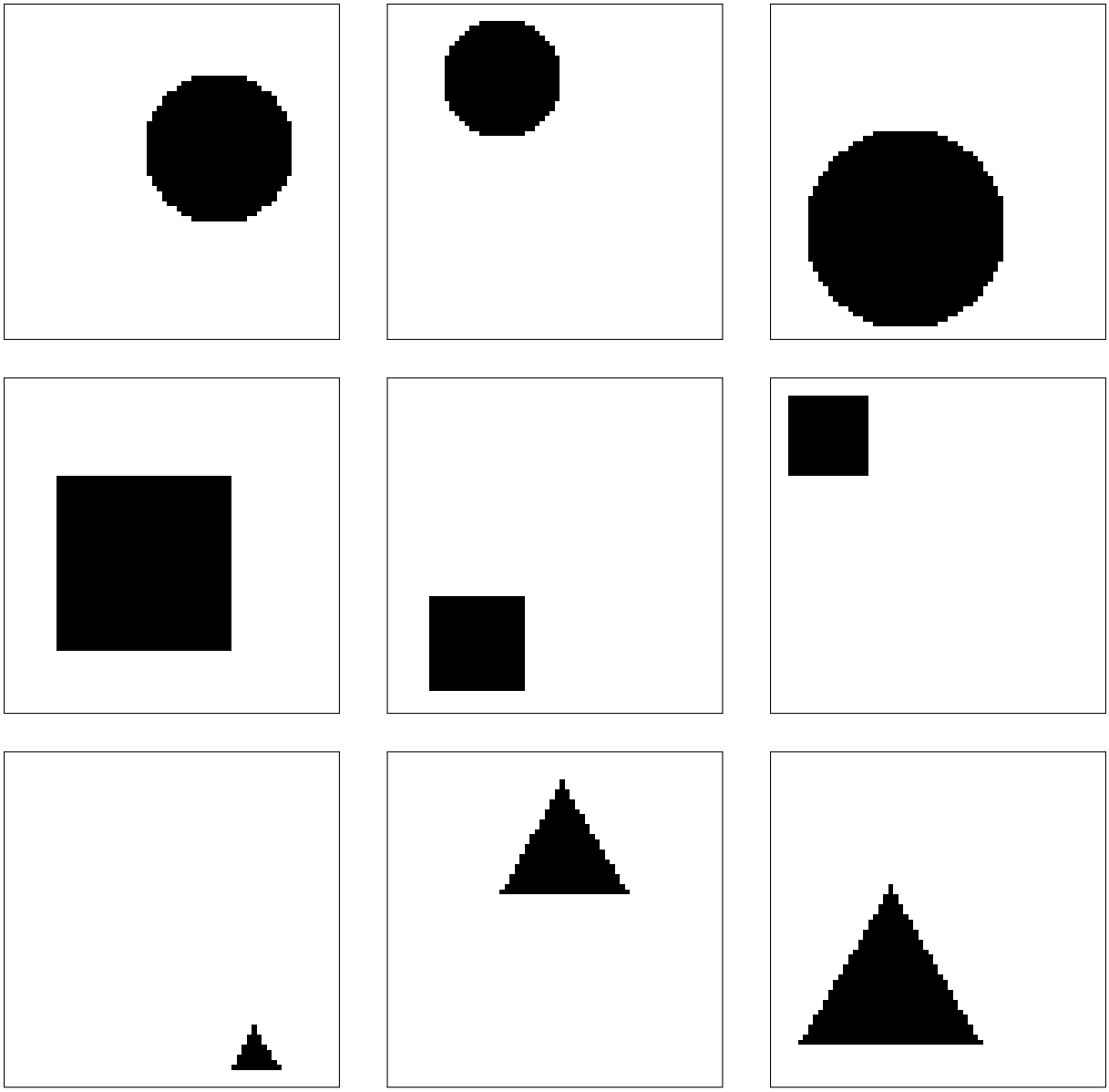

Abbildung 3.1: Auswahl an zufällig generierten Trainingsbilder

Größe

Alle Bilder benötigen die gleiche Größe, um zur späteren GAN-Architektur zu passen. Damit die geometrischen Formen auch erkennbar sind, dürfen sie jedoch nicht zu klein sein, wohingegen zu große Bilder Trainingszeiten unnötig verlängern. Die Bilder sind 64x64 Pixel groß, was einen Kompromiss darstellt.

3.1.2 Eigenschaften der Figuren

Die Trainingsbilder für das GAN stellen jeweils eine geometrische Form dar. Bei den Formen handelt es sich um Kreise, Dreiecke und Quadrate, die von Bild zu Bild unterschiedlich sind. Die Bilder unterscheiden sich in mehreren Aspekten:

Position

Die Formen sind zufällig auf dem Bild positioniert. Allerdings sind sie immer vollständig abgebildet, das heißt, die Kanten des Bildes schneiden die Form nicht. Je nach Größe der Form muss so die Position treffend gewählt werden.

Größe und Form

Die Formen sind unterschiedlich groß, besitzen aber eine minimale und maximale Größe. So ist gewährleistet, dass Bilder nicht einfarbig erscheinen und die Form immer erkennbar bleibt. Die Minimalgröße beträgt bei den Formen 10 Pixel und Maximalgröße 32 Pixel. Die Figuren werden bei unterschiedlichen Größen unter gleichbleibenden Seitenverhältnissen skaliert.

3.2 Training

Im Trainingsprozess sollen verschiedene Architekturen mit unterschiedlichen Hyperparametern getestet werden. Die ausgewählten Architekturen orientieren sich an bereits existierender Forschung zu möglichst allgemeingültigen Architekturen (siehe Abschnitt 2.3). Für jede Architektur wird außerdem eine Hyperparameteroptimierung von ausgewählten Hyperparametern durchgeführt. Dadurch ist gewährleistet, dass tatsächlich die Architektur und nicht die Hyperparameter ausschlaggebend für die Ergebnisse sind.

3.2.1 Auswahl zu optimierender Hyperparameter

Es ist nicht möglich, alle möglichen Hyperparameter zu optimieren. Eine solche Optimierung benötigt zu viele Ressourcen, die im Rahmen der Studienarbeit nicht vorhanden sind. Es wurde deshalb eine Vorauswahl getroffen, die im Folgenden erläutert wird.

Erläuterung hinzufügen

- Learning-Rate Generator
- Learning-Rate Discriminator
- Dropout Discriminator
- NumUnits in bestimmten Layern

3.3 Erfolgskriterien

Damit die Konfigurationen treffend vorausgewählt werden könne, müssen die Kriterien für ein erfolgreich trainiertes GAN klar definiert werden. Als Erfolgskriterien zählen in diesem Fall die Varietät der generierten Bilder und die Korrektheit der generierten Figuren. Beide Kriterien werden im Folgenden noch einmal genauer erläutert.

Varietät

Die Varietät bezieht zum einen auf die Ähnlichkeit zwischen Trainingsbildern und den generierten Bildern der GANs. Die Ähnlichkeit zwischen diesen beiden Bildersets beschreibt, wie gut das GAN 'etwas neues schaffen' kann, oder ob es nur Trainingsbilder dupliziert. Sollte die Ähnlichkeit sehr gering sein, werden viele 'neue' Bilder generiert, was sehr positiv zu bewerten ist. Zudem bezieht sich die Varietät auf die generierten Bilder untereinander. Sie sollten auch möglichst verschieden sein, was oftmals nicht der Fall ist. Das Phänomen ist als 'mode-collapse' bekannt (siehe Stand der Technik).

Beide Probleme lassen sich durch den Vergleich der Bilder mittels 'Structural Similarity Index' [28] bewerten.

Korrektheit

Neben der Varietät besitzt auch die Korrektheit eine hohe Bedeutung für die Bewertung der GANs. Dabei muss sichergestellt werden, dass die richtige Figur generiert wurde und erkennbar ist. Die Figuren müssen außerdem den gleichen Kriterien wie die Trainingsdaten genügen, das heißt, die Figuren sollten zum Beispiel vollständig innerhalb des Bildes abgebildet sein. Es ist allerdings eher unwahrscheinlich, dass das GAN Bilder generiert, die keinem Pendant aus den Trainingsbildern entsprechen. Ein weiterer wichtiger Aspekt der Bilder ist das Verhalten im Hintergrund der Figur. So sollten im Hintergrund möglichst keine anderen Figuren oder Pixelfragmente erzeugt werden.

All diese Kriterien werden durch eine manuelle Überprüfung evaluiert werden.

- Inception Score zum Rating von erzeugten Bildern (Salimans et al. 2016)
- Frechet Inception Distance (Heusel et al. 2017)

3.4 Verfahren zur Hyperparametersuche

Zur Auswahl von Hyperparametern gibt es verschiedene Verfahren. Dazu gehören unter anderem die Manuelle Suche, Gridsearch, Random Search und Genetic Algorithms. Die Verfahren sind jeweils im Stand der Technik erklärt (siehe Abschnitt 2.2). In diesem Abschnitt werden die verschiedenen Verfahren verglichen und zusätzliche Anpassungen erläutert.

3.4.1 Vergleich

In der Studienarbeit wird ausschließlich die Gridsearch zur Bestimmung der Hyperparameter zur Anwendung kommen. Die Entscheidung ist mit den nachfolgenden Punkten begründet.

Manuelle Suche

Die ausschließlich manuelle Suche ist zwar sehr flexibel, aber zu aufwändig. Eine weitreichende Suche mit ständiger Neuevaluierung der Trainingsergebnisse ist für den kurzen Zeitraum der Studienarbeit nicht verhältnismäßig. Zudem ist das Verfahren für eine komplexe Hyperparametersuche nicht mehr zeitgemäß. Die anderen vorgestellten Verfahren sind der manuellen Suche immer überlegen.

anderes wort

Zufallssuche

Die Zufallssuche ist der Gridsearch sehr ähnlich. Statt festgelegten Zahlen werden Zufallszahlen verwendet. Dadurch wird ein größeres Spektrum an Kombinationen abgedeckt. Allerdings führt das Verfahren in der Praxis nicht unbedingt zu besseren Ergebnissen als die Gridsearch. Zusätzlich führt es zu einem größeren Implementierungsaufwand und wieder mehr Hyperparametern als die Gridsearch. Der erhöhte Implementierungsaufwand ist durch eine zusätzliche Zufallsfunktion zu begründen. Die Funktion wird benötigt, um die alle Kombinationen anzupassen. Im Vergleich dazu kann bei der Gridsearch pro Hyperparameter ein Array mit verschiedenen Werten übergeben werden. Die zusätzlichen Hyperparameter hängen auch mit der Zufallsfunktion zusammen. Denn der zufällige Abstand zum Originalwert muss in einem sinnvollen Intervall liegen. Die Bestimmung der Grenzen des Intervalls benötigt zusätzliche Anpassungen und Training.

Evolutionäre Suche

Die evolutionäre Suche ist sowohl mit einem hohen Implementierungs-, als auch Rechenaufwand verbunden. Dafür ist es möglich, sehr viele Hyperparameter auf ihre Zusammenhänge zu untersuchen. Jedoch reduzieren wir die Hyperparameter auf eine kleine Auswahl, weswegen der Aufwand nicht gerechtfertigt ist.

Gridsearch

Die Gridsearch ist für die Arbeit am besten geeignet, da sie zum einen einen sehr geringen Implementierungsaufwand mit sich bringt. Es müssen nur die Hyperparameter festgelegt werden, der Suchprozess kann dann mithilfe von Tensorboard grafisch veranschaulicht werden.

3.4.2 Anpassung an Gridsearch

Zwar findet die Gridsearch in keiner intelligenten Weise die optimalen Hyperparameter, aber durch kurze Trainings- und Validierungsintervalle können schlecht trainierende Kombinationen schnell aussortiert werden. Dafür können am Anfang n GANs auf m Generationen trainiert werden. Nach dem Training werden die Ergebnisse evaluiert und die Hälfte der GANs aussortiert Im Gegenzug wird die Anzahl der zu trainierenden Generationen verdoppelt. So bleibt der Aufwand für das Training, bei deutlich erhöhter Intensität, der gleiche. Es ist anzumerken, dass die neuronalen Netze weitertrainiert werden und der vorhandene

Lernfortschritt nicht zurückgesetzt wird. Der Prozess wird beliebig oft wiederholt und es können zu jedem Zeitpunkt Zusammenhänge zwischen den verbliebenen GANs analysiert werden.

4 Implementierung

4.1 Umgebung

- Python Version
- Tensorflow Version (+ CUDA)
- Keras Version
- pip-install erwähenen ? → wie setzt man das Projekt auf?
- github

4.2 Model

- Aufbau Generator + Discriminator (Bild)
- Hyperparameter, die wir so gesetzt haben (Dropout, LeakyReLU, ?)

4.3 Generierung von Trainingsdaten

- Datei zum Konfigurien erklären
- Erklärung, was da drin ist (Datensatz-Generierung)

5 Ergebnisse

6 Diskussion

7 Zusammenfassung

Literatur

[1] tbd. *Generative Adversarial Nets*. 0. URL: https://arxiv.org/pdf/1406.2661.pdf (besucht am 29. 01. 2022).

[2] tbd. *GAN Augmentation: Augmenting Training Data using Generative Adversarial Networks*. 0. URL: https://arxiv.org/abs/1810.10863 (besucht am 29. 01. 2022).

[3] tbd. *High-resolution image de-raining using conditional GAN with sub-pixel upscalingl)*. 0. URL: https://link.springer.com/article/10.1007/s11042-020-09642-7 (besucht am 29. 01. 2022).

[4] tbd. *BlendGAN: Implicitly GAN Blending for Arbitrary Stylized Face Generation*. 0. URL: https://proceedings.neurips.cc/paper/2021/file/f8417d04a0a2d5e1fb5c5253a3656
Paper.pdf (besucht am 29. 01. 2022).

[5] tbd. *Feedback GAN for DNA optimizes protein functions*. 0. URL: https://www.nature.com/articles/s42256-019-0017-4 (besucht am 29. 01. 2022).

[6] tbd. *Fre-GAN: Adversarial Frequency-consistent Audio Synthesis*. 0. URL: https://arxiv.org/abs/2106.02297 (besucht am 29. 01. 2022).

[7] Thalles Silva. *An intuitive introduction to Generative Adversarial Networks (GANs)*. 0. URL: https://www.freecodecamp.org/news/an-intuitive-introduction-to-generative-adversarial-networks-gans-7a2264a81394/ (besucht am 29. 01. 2022).

[8] tbd. *Ian Goodfellow: Generative Adversarial Networks (NIPS 2016 tutorial)*. 0. URL: https://youtu.be/HGYYEUSm-0Q?t=2004 (besucht am 29. 01. 2022).

[9] tbd. *A (conditional) Generative Adversarial Network*. 0. URL: https://www.mathematik.uni-wuerzburg.de/fileadmin/10040900/2021/A__conditional__Generative_Adversarial_Network_BastianDittrich__LINKS.pdf (besucht am 29. 01. 2022).

[10] Towards Data Science. *Understanding Hyperparameters and its Optimisation techniques*. 0. URL: https://towardsdatascience.com/understanding-hyperparameters-and-its-optimisation-techniques-f0debba07568 (besucht am 29. 01. 2022).

[11] Fajr Ibrahem Alarsan, Mamoon Younes. *Best Selection of Generative Adversarial Networks Hyper-Parameters using Genetic Algorithm*. 0. URL: https://assets.researchsquare.com/files/rs-95571/v1_stamped.pdf (besucht am 29. 01. 2022).

[12] Y. Bengio. „Practical recommendations for gradient-based training of deep architectures". In: Bd. 7700. 2012, S. 437–478.

[13] tbd. *Automated Machine Learning*. 0. URL: https://link.springer.com/content/pdf/10.1007%2F978-3-030-05318-5.pdf (besucht am 29. 01. 2022).

[14] tbd. *Random Search for Hyper-Parameter Optimization*. 0. URL: https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a?ref=https://githubhelp.com (besucht am 29. 01. 2022).

[15] David Wipf und Srikantan Nagarajan. „A New View of Automatic Relevance Determination". In: *Advances in Neural Information Processing Systems*. Hrsg. von J. Platt u. a. Bd. 20. Curran Associates, Inc., 2008. URL: https://proceedings.neurips.cc/paper/2007/file/9c01802ddb981e6bcfbec0f0516b8e35-Paper.pdf.

[16] tbd. *Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS*. 0. URL: https://arxiv.org/pdf/1912.06059.pdf (besucht am 29. 01. 2022).

[17] tbd. *Best Selection of Generative Adversarial Networks Hyper-Parameters Using Genetic Algorithm*. 0. URL: https://sci-hub.se/10.1007/s42979-021-00689-3 (besucht am 29. 01. 2022).

[18] tbd. *Classifier Systems and Genetic Algorithms*. 0. URL: https://sci-hub.se/10.1016/0004-3702(89)90050-7 (besucht am 29. 01. 2022).

[19] tbd. *Automated Machine Learning*. 0. URL: http://yann.lecun.com/exdb/mnist/ (besucht am 29. 01. 2022).

[20] tbd. *UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS*. 0. URL: https://arxiv.org/pdf/1511.06434.pdf%C3%AF%C2%BC%E2%80%B0 (besucht am 29. 01. 2022).

[21] Fisher Yu u. a. „LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop“. In: *arXiv preprint arXiv:1506.03365* (2015).

[22] tbd. *ImageNet: A large-scale hierarchical image database*. 0. URL: `https://ieeexplore.ieee.org/document/5206848` (besucht am 29. 01. 2022).

[23] tbd. *THE GAN LANDSCAPE: LOSSES, ARCHITECTURES, REGULARIZATION, AND NORMALIZATION*. 0. URL: `https://openreview.net/pdf?id=rkGG6s0qKQ` (besucht am 29. 01. 2022).

[24] tbd. *The CIFAR-10 dataset*. 0. URL: `https://www.cs.toronto.edu/~kriz/cifar.html` (besucht am 29. 01. 2022).

[25] Smeschke, Hrsg. *Four Shapes (16.000 iamges of four basic shapes (star, circle, square, triangle)*. 21. Nov. 2017.

[26] Youssef Ghanou Anas El Korchi. „2D geometric shapes dataset for machine learning and pattern recognition“. Data Article. University Moulay Ismail of Meknes, Marocco, 5. Juli 2020.

[27] Mario Lucic, Karol Kurach, Marcin Michalski, Olivier Bousquet, Sylvain Gelly. „Are GANs Created Equal? A Large-Scale Study“. Google Brains, 29. Okt. 2018.

[28] Zhou Wnag, Alan Conrad Bovik, Hamid Rahim Sheikh, Eero P. Simoncelli. „Image Quality Assessment: From Error Visibility to Structural Similarity“. IEEE, 4. Apr. 2004.