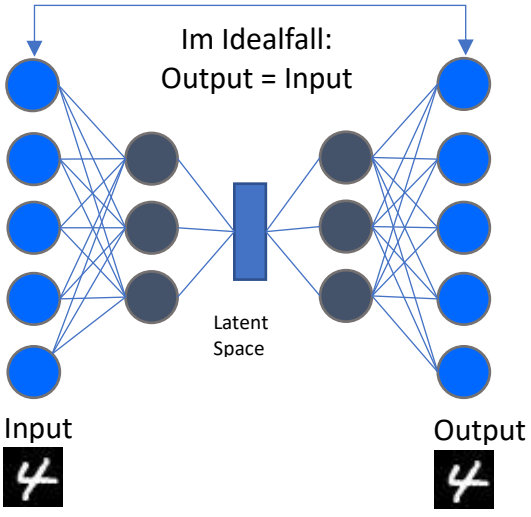
Fynn Madrian

Image Reconstruction mit Autoencodern und GANs

In der heutigen Ära der digitalen Bildverarbeitung und -reproduktion stellen Bildstörungen und Artefakte eine allgegenwärtige Herausforderung dar. Diese Artefakte können durch verschiedene Faktoren entstehen, wie beispielsweise eine schlechte Bildqualität, Übertragungsfehler oder unzureichende Aufnahmetechniken. Solche Störungen können das visuelle Erscheinungsbild von Bildern stark beeinträchtigen und ihre Nutzungswerte mindern. Daher besteht ein wachsender Bedarf an leistungsstarken Techniken zur Wiederherstellung von Bildern mit Artefakten.

Ein häufig in Bildern auftretendes Artefakt ist Rauschen, also unerwünschte Variation von Farbe und Helligkeit. Dieses ist sehr ähnlich zu statischem Rauschen im Audiobereich, welches beispielsweise bei schlechtem Radioempfang auftritt. Bei Bildern entsteht Rauschen sehr häufig bei schlechten Lichtverhältnissen oder falschen Kameraeinstellungen. Um dieses Rauschen mithilfe eines neuronalen Netzes zu entfernen, muss das Modell in der Lage sein, den Inhalt des Bildes vom Rauschen zu unterscheiden, um nur das Rauschen entfernen zu können. Eine mögliche Modellstruktur wäre hierfür ein Autoencoder [1]. Dieser besteht aus 2 Teilen, einem Encoder und einem Decoder. Ziel des Encoders ist es, das gesamte Bild in den sogenannten Latent Space zu überführen. Der Latent Space ist eine kleinere Abbildung der Originaldaten. Aus all den Pixeln im Original wird hier eine möglichst genaue Beschreibung in sehr wenigen Zahlenwerten, bei der möglichst wenige Information verloren gehen soll. Der Decoder benutzt diesen Latent Space als Input, um das Bild wieder herzustellen. Der Output des Decoders kann dann mit dem Originalbild verglichen werden, um den Fehler zu berechnen und somit auch um das Netzwerk zu trainieren.

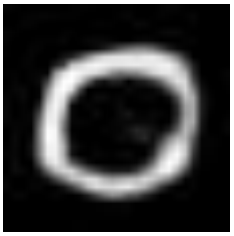
Struktur eines Autoencoders:



Im ersten praktischen Versuch wurde hierzu der bekannte MNIST-Datensatz [2] verwendet. Dieser Datensatz beinhaltet 60.000 Bilder von handgeschriebenen Ziffern und wird häufig als Beispiel für Klassifikationsalgorithmen verwendet. Normalerweise sollen die Bilder mit Hilfe von einfacheren Modellen den korrespondierenden Zahlen zugeordnet werden. In diesem Fall werden die Daten jedoch nicht zur Klassifikation verwendet. Die Label werden somit auch nicht verwendet und werden verworfen. Stattdessen bekommt das Modell die Originalbilder mit einer Auflösung von 28 mal 28 Pixeln als Input, überführt diese in eine kleinere Repräsentation aus 32 Zahlenwerten und kann daraus das Bild rekonstruieren. Der Output wird dann mit dem Input verglichen, und mit dem Fehler das Modell angepasst. Das wird so lang wiederholt, bis das Modell ausreichend trainiert ist und die Outputs möglichst gleich zum Input sind. Das Modell versucht im Training, Strukturen und Muster im Trainingsdatensatz möglich effizient in den Latent Space zu codieren, sodass diese dann im Decoder wieder zum Gesamtbild rekonstruiert werden können. Je effizienter diese Muster im Latent Space vorliegen, desto besser sieht am Ende auch das Ergebnis aus.

Wenn das Netzwerk nun ein verrauschtes Bild verarbeitet und das Bild in den Latent Space übertragen wird, kann das Rauschen nicht codiert werden, da es nicht im Training enthalten war. Das Rauschen besitzt kein Muster, mit welchem es vereinfacht werden könnte, und kann dadurch vom Encoder nicht mit in den limitierten Bereich des Latent Space übertragen werden. Somit geht diese Information verloren, und das vom Decoder erzeugte Bild enthält kein Rauschen. Durch diesen Prozess können zwar auch Informationen des tatsächlichen Objektes verloren gehen, die Zahl ist jedoch immer noch deutlich erkennbar und dafür vollkommen rauschfrei. Indem das Training auch mit unterschiedlichen Stärken von Rauschen durchgeführt wird, lässt sich die Robustheit des Models ebenfalls verbessern, wodurch die rekonstruierten Zahlen deutlich besser leserlich sind. Die folgenden Ergebnisse wurden auf Testdaten erzielt:

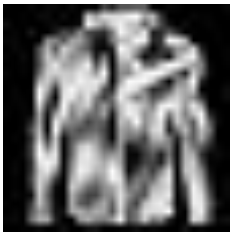

verrauscht rekonstruiert

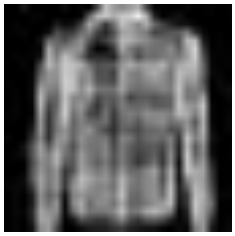



verrauscht rekonstruiert

Im nächsten Schritt wurde der MNIST Fashion Datensatz [3] verwendet, da diese Bilder die gleiche Auflösung wie der reguläre MNIST Datensatz besitzen und ebenfalls schwarz-weiß sind. Der entscheidende Unterschied hier ist, dass der Fashion Datensatz aus deutlich unterschiedlicheren Bilder besteht, die zudem auch visuell komplexer sind.

Zunächst wurde dieselbe Modellstruktur wie im vorherigen Versuch verwendet, nur mit dem neuen Datensatz. Die nach dem Training rekonstruierten Bilder waren zwar größtenteils rauschfrei, allerdings waren die Ergebnisse an sich nicht mit den Originalen vergleichbar. An vielen Stellen waren Löcher in den Objekten und besonders dichte Flecken an Rauschen wurden nicht entfernt, sondern stattdessen sogar verstärkt. Um diese Problematik zu umgehen wurde statt einem herkömmlichen Autoencoder im weiteren Verlauf ein denoising Autoencoder [4] verwendet. Während ein herkömmlicher Autoencoder darauf abzielt eine möglichst genaue Rekonstruktion der Eingabedaten zu erzeugen, versucht ein denoising Autoencoder eine saubere Version der ursprünglichen Daten zu rekonstruieren indem er das Rauschen unterdrückt. Dies wird erreicht, indem den Eingabedaten absichtlich Rauschen hinzugefügt wird. Der denoising Autoencoder wird dann darauf trainiert das ursprüngliche Bild ohne Rauschen auszugeben. Dadurch lernt das Modell relevante Merkmale der Daten zu extrahieren und das Rauschen zu unterdrücken. Dies führt zu einer verbesserten Datenrekonstruktion. Konkret wird also im Training bereits ein verrauschtes Bild als Input verwendet und der Output wird mit dem Originalbild verglichen. Das Rauschen wird somit direkt im Lernprozess integriert. Mit dieser Anpassung, kombiniert mit einem vergrößerten Latent Space, war es dem denoising Autoencoder möglich die komplexeren Fashion-MNIST Bilder erfolgreich zu rekonstruieren, selbst mir sehr starkem Rauschen. Wichtig hier ist, dass das Modell verlorene Informationen nur bedingt wiederherstellen kann. Somit werden stark verrauschte Bereiche mit dem wahrscheinlichsten Ergebnis gefüllt. Das rekonstruierte Bild sieht somit im Vergleich zum verrauschten Bild deutlich besser aus, weißt aber im Vergleich zum Original große Unterschiede auf. Das Modell versucht hauptsächlich das Rauschen zu entfernen. Wenn dieses jedoch besonders stark ist und somit Bereiche des Bildes vollkommen verdeckt, kann das Modell diese zwar trotzdem vom Rauschen bereinigen, jedoch sind die Ergebnisse nicht unbedingt mit dem Original vergleichbar.





Original verrauscht rekonstruiert

Wie sich gezeigt hat, ist besonders die Rekonstruktion von konzentrierterem Rauschen ein Problem. Um diesen Informationsverlust effizienter zu rekonstruieren, sollte dieses Element bereits konkret im Training enthalten sein. Eine Möglichkeit ist es an einer zufälligen Position im Bild ein großes Artefakt zu platzieren. In diesem Fall werden an dieser Stelle alle Farbkanäle auf das Maximum gesetzt, wodurch ein weißes Rechteck auf dem Bild entsteht. Diese Artefakte werden während dem Training dynamisch auf die Bilder gelegt. Somit entsteht eine höhere Varianz im Datensatz im Austausch für eine leicht erhöhte Trainingsdauer durch den zusätzlichen Rechenaufwand. Mit diesem neuen Datensatz wurde das vorher beschriebene Modell erneut von Grund auf trainiert. Mit minimaler Optimierung war das Modell bereits in der Lage, die eingefügten Artefakte zu entfernen und das Originalbild überzeugend zu rekonstruieren. Es besteht zwar ein sichtbarer Unterschied zum originalen Bild, dieser ist aber nur im direkten Vergleich sichtbar. Das rekonstruierte Bild würde alleinstehend unter einer Sammlung an Originalen vermutlich nicht auffallen.

Links: Original Mitte: mit Artefakt Rechts: rekonstruiert

Diese simplen Anwendungen zur Bildrekonstruktion dienen mehr als Proof-of-Concept als für eine wirkliche Nutzung. In der Realität sind Bilder nicht nur schwarz-weiß und auch selten fotografieren Leute massenhaft handgeschriebene Ziffern, um diese mit anderen zu teilen. Deutlich nützlicher ist die Rekonstruktion von echten Fotografien, welche man auch mit Freunden und Familie teilen würde.

Optimal für diesen Nutzungsfall hat sich der Flickr-Faces-High-Quality [5] Datensatz angeboten. Der Datensatz enthält 70.000 Bilder mit einer Auflösung von 1024 mal 1024 die Nahaufnahmen von Gesichtern zeigen. Eine etwas handlichere Bildgröße wäre jedoch die hier verwendete Miniversion des Datensatzes mit einer Auflösung von 128 auf 128. Durch die größere Bildauflösung und dem Zusatz von zwei extra Farbkanälen muss die Modelstruktur ebenfalls dementsprechend angepasst werden. Da nun deutlich mehr Information in den Bildern enthalten ist, musste sowohl die Größe des Latent Spaces angepasst werden, als auch die Tiefe des Modells. Der Autoencoder, der für diese Bilder verwendet wurde, hatte je vier Convolution- und Deconvolutionblöcke, wodurch der Latent Space insgesamt aus 1024 Werten bestand. Das entspricht ungefähr 2% der Größe des Inputs. Das tiefere Modell wurde mit dem neuen Datensatz trainiert, welches mit den gleichen Artefakten wie im vorherigen Beispiel versehen wurden.

Die mit diesem Modell rekonstruierten Bilder sind sehr verwischt und enthalten wenige Details. Die Gesichter sind zwar noch erkennbar, aber im Vergleich zu den Originalen sichtlich verallgemeinert. Die Artefakte sind immer noch deutlich zu sehen, auch wenn sie teilweise durch bunte Farbflecken ersetzt wurden. Mit viel Glück sehen einzelne Bilder akzeptabel aus, allerdings nur wenn das Artefakt zufällig an einer günstigen Stelle gelandet ist und das Modell das Bild nicht vollkommen unkenntlich gemacht hat.

Entrauschte Bilder:

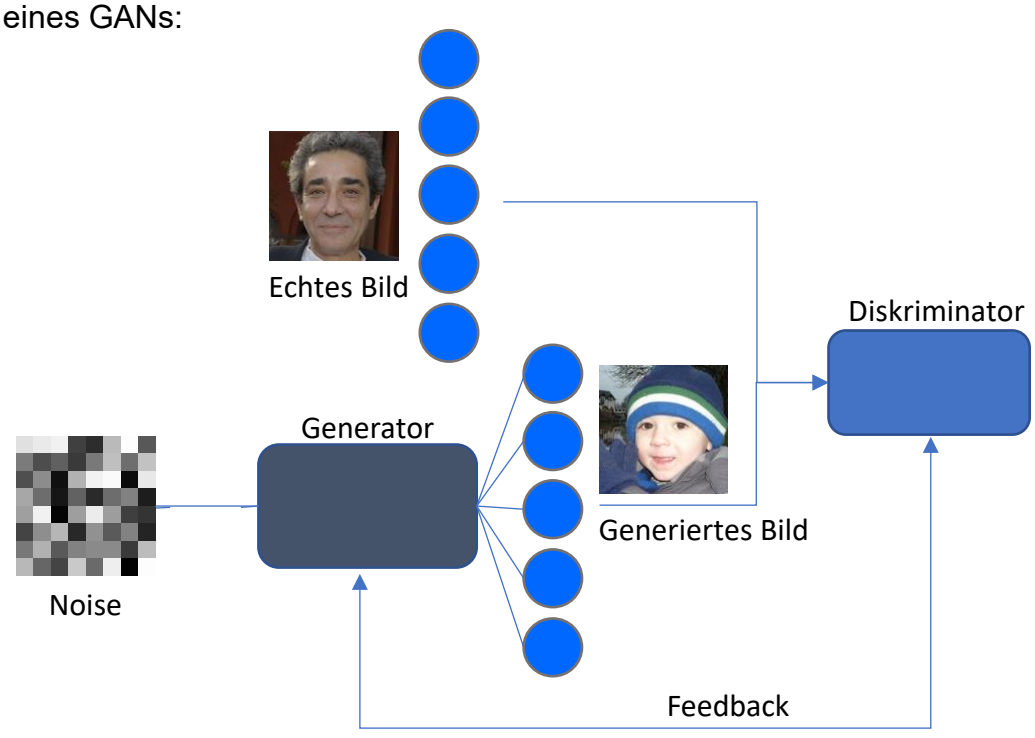
Nach einigen Trainingsiterationen und mehreren Anläufen ließ sich das Ergebnis zwar etwas verbessern, das visuelle Ergebnis war allerdings immer noch sehr unscharf und das Artefakt war in fast allen Bildern noch deutlich erkennbar.

Trotz der Fortschritte in der Optimierung gibt es ein inhärentes Limit, welches verhindert, dass das Modell unendlich optimiert werden kann. Dieses Limit liegt in der

Natur des Lernprozesses selbst. Das Modell passt iterativ seine internen Parameter an um den Fehler zu minimieren. Ab einem bestimmten Punkt ist ein Minimum erreicht, das das Modell mit dieser Struktur nicht unterschreiten kann. Somit hat dieses Modell für diesen Anwendungsfall zumindest die Nähe des Limits erreicht, ohne das Problem befriedigend lösen zu können. Es wird also eine neue Modellstruktur benötigt.

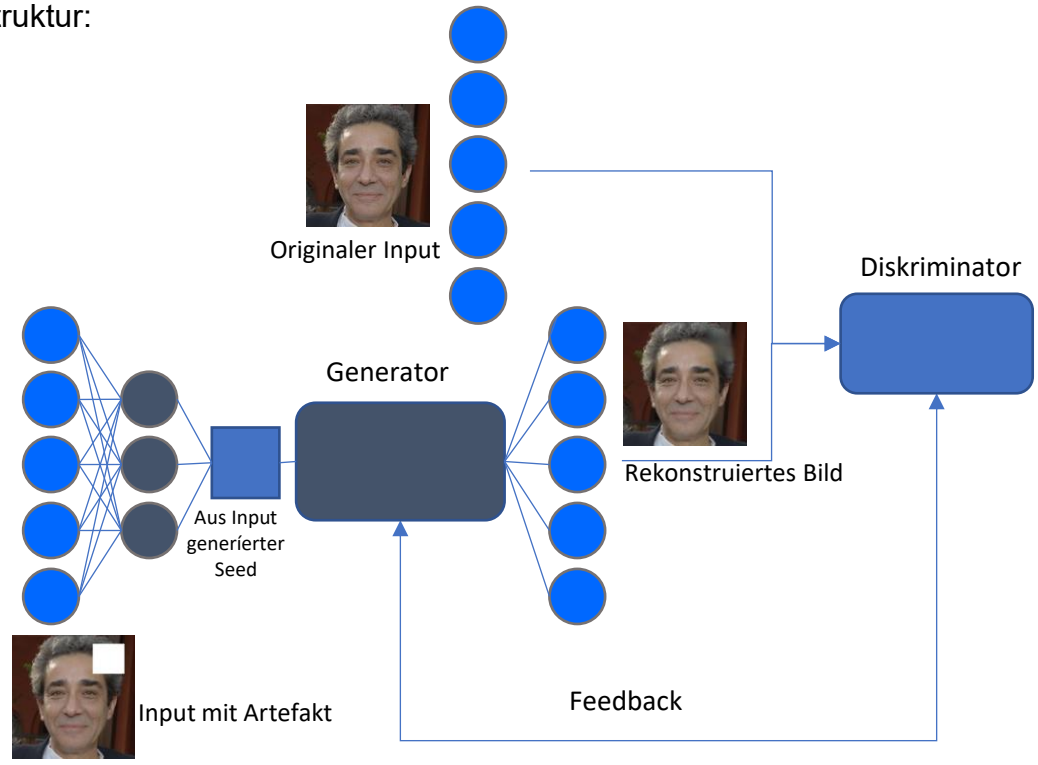
Das neue Modell ist ein sogenanntes Generative Adversarial Network [6], kurz auch GAN genannt. Das Modell besteht, ähnlich wie der Autoencoder, erneut aus zwei Komponenten: dem Generator und dem Diskriminator. Diese haben hier allerdings vollkommen andere Aufgaben als im Autoencoder. Der Generator verwendet einen sogenannten Seed, im Normalfall Zufallswerte, um daraus ein Bild zu generieren welches denen im Trainingsdatensatz möglichst stark ähneln soll. Diese generierten Bilder werden dann zusammen mit echten Bildern an den Diskriminator übergeben, welcher im Anschluss entscheiden muss, ob die Bilder echt oder generiert sind. Somit arbeiten die beiden Teile des Modells gegeneinander: Während der Generator versucht, immer bessere Fälschungen zu generieren, wird der Diskriminator zunehmend besser die Fälschungen von echten Bildern zu unterscheiden. Sie sind dadurch im Training voneinander abhängig, was das Training stark komplizieren kann. Um das Modell effizient trainieren zu können, muss ein stabiles Equilibrium zwischen den beiden Teilen des Models bestehen. Dies kann oft ein Problem darstellen, da der Generator und der Diskriminator ständig versuchen sich gegenseitig zu übertreffen, wodurch oft Instabilität und Divergenz entsteht.

Struktur eines GANs:

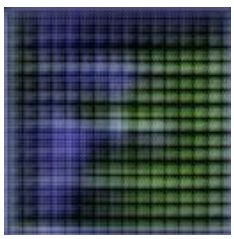


Ein standardmäßiges GAN verwendet für den Input in den Generator Zufallswerte, um vollkommen neue Bilder zu generieren. Für diesen Anwendungsfall soll das generierte Bild allerdings eine Rekonstruktion des Originalbildes sein und muss somit im Input enthalten sein. Um das gewünschte Ergebnis zu erzielen, muss das Originalbild also erst mithilfe von mehreren Convolutionallayern in eine niedriger dimensionale Repräsentation überführt werden. Diese Repräsentation kann dann als Input für den Generator verwendet werden.

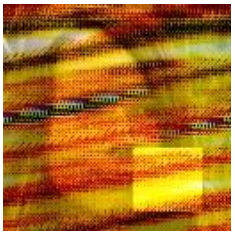
Angepasste Struktur:



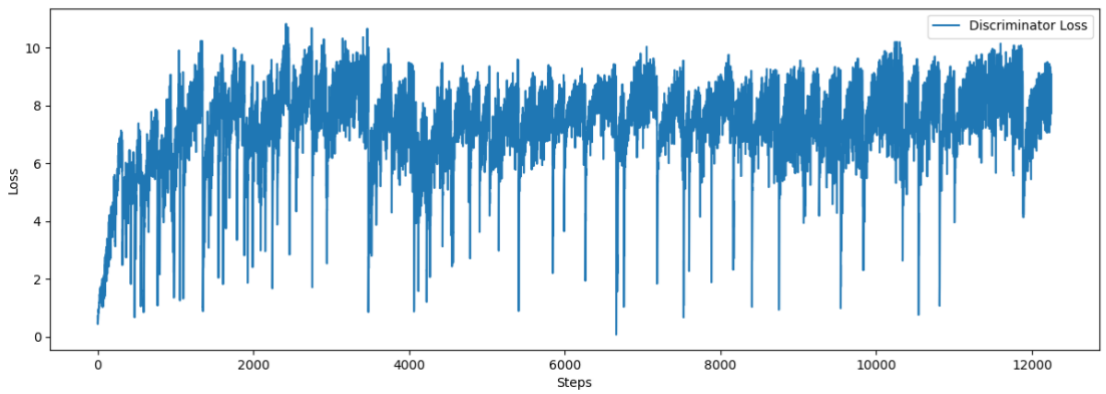
Es war insgesamt sehr schwierig das GAN korrekt aufzusetzen und selbst nach einiger Arbeit waren die Ergebnisse sehr schwach. Auch nach langem Training waren die Bilder kaum als Gesichter zu erkennen und sahen teilweise sehr gruselig aus. Selbst ohne die Artefakte war es dem Modell nicht wirklich möglich Gesichter zu generieren. Mit viel Interpretation sind Augen oder andere gesichtsähnliche Strukturen zu erkennen. Insgesamt bestehen die Bilder jedoch hauptsächlich aus verschwommenen Farbverläufen.



Auch nach weiterer iterativer Optimierung der Ergebnisse war der Großteil der Ergebnisse noch eine Mischung aus Farben mit Ansätzen von gesichtsähnlichen Strukturen. Selbst nach langer Trainingsdauer waren die besten Bilder lediglich die Inputbilder, welche mit noch mehr Rauschen und Farbschmierern zusätzlich zerstört wurden.



Das Training von GANs tendiert oft dazu zu divergieren. Der Fehler des Modells steigt also systematisch über den Trainingsverlauf, anstatt nach und nach kleiner zu werden. Das Modell wird also aktiv immer schlechter, auch wenn es zuvor sogar in der Lage war akzeptable Ergebnisse zu produzieren. Der folgende Graph zeigt den Trainingsverlauf des Diskriminators. Die X-Achse zeigt die Trainingssteps, also den zeitlichen Verlauf, und die Y-Achse zeigt den Fehler (hier als Loss betitelt). Je geringer der Fehler, desto besser kann das Modell laut der Fehlermetrik das gegebene Problem lösen. Im Idealfall sollte der Fehler über den Verlauf des Trainings kontinuierlich abnehmen. In diesem Fall ist der Verlauf allerdings sehr sprunghaft, besonders der des Diskriminators. Der Verlauf des Generators ist etwas weniger sprunghaft, scheint allerdings nach einer Weile nicht weiter zu sinken, da das Feedback des Diskriminators nichtmehr aussagekräftig ist. Dieses Verhalten ist beim Training von GANs üblich, da diese Modelstruktur sehr sensibel auf Hyperparameter ist und sehr genau konfiguriert werden muss.



Da der Diskriminator mit diesem Ansatz nur um einen Zustand zu schwingen scheint anstatt sich über den Trainingsverlauf zu verbessern, war auch der Generator nicht in der Lage aus dem Feedback zu lernen. Auch nach mehreren Trainingsansätzen sahen die Ergebnisse ähnlich aus. Das Modell scheint sich anfangs zu verbessern, doch bereits früh entstehen keine wirklichen Veränderungen mehr. Anstatt also diesen fehlerbehafteten Ansatz weiter zu optimieren, muss stattdessen ein stabilerer Ansatz gefunden werden, mit welchem das Modell konvergieren kann.

Der erste Durchbruch wurde durch eine Erweiterung des Modells und durch eine andere Trainingsmethode erzielt. Die neue Struktur wurde von einem Open-Source-Projekt [7] zur Wiederherstellung verwischter Bilder inspiriert und Teile des Codes angepasst übernommen

Häufig stellt das Training von tieferen und komplexeren Netzwerken ein Problem dar, welches sich hier durch die Implementierung von ResNet-Blöcken [8] umgehen ließ. Ein ResNet-Block besteht aus einer Sequenz von Convolutional Layers, die von einer sogenannten Residual Connection begleitet werden. Die Residual Connection ermöglicht es die Ausgabe eines Convolutional Layers direkt zur Eingabe des Blocks hinzuzufügen. Anstatt die Eingabe des Blocks durch den Convolutional Layer komplett zu ersetzen, wird sie aufaddiert. Dieser Mechanismus erlaubt dem Netzwerk eine Residual-Funktion zu erlernen, die den Unterschied zwischen der Ausgabe und der Eingabe des Blocks repräsentiert. Die Residual Connection bringt einige entscheidende Vorteile für das Training von GANs mit sich. Beispielsweise hilft sie dabei das Problem des vanishing bzw. exploding Gradients zu überwinden. Durch die Residual Connection wird der Gradient, der durch die Schichten des Blocks fließt, direkt zur Eingabe weitergeleitet. Das ermöglicht eine einfachere Backpropagation des Gradienten und erleichtert somit das Training des GANs.

Ein weiterer Vorteil der Residual Connection besteht darin, dass sie das Lernen von tiefen Modellen erleichtert. Tiefere GAN-Architekturen haben das Potenzial hochwertigere und realistischere Ergebnisse zu erzeugen, da sie komplexe Muster und Strukturen erfassen können. Durch die Residual Connection wird es einfacher solche tiefen Architekturen zu trainieren, da sie den Gradientenfluss stark erleichtert.

Das Modell kann ebenfalls durch Trainingsparameter optimiert werden, ohne die Struktur des Modells anzupassen. Diese Parameter sind bei GANs besonders relevant, da der Generator und der Diskriminator sowohl miteinander als auch gegeneinander agieren. Es muss also im Training eine gute Balance gefunden werden, da bei einem schlechten Diskriminator der Generator sehr wenig relevantes Feedback erhält und somit kaum lernen kann. Auf der anderen Seite kann sich bei einem zu guten Generator der Diskriminator kaum anpassen. Der wichtigste Parameter war in diesem Fall, wie häufig der Diskriminator im Vergleich zum Generator trainiert wird. Anstatt beide gleich oft mit den Trainingsbildern anzupassen, wird der Diskriminator fünfmal so oft mit echten und rekonstruierten Bildern trainiert wie der Generator.

Da die rekonstruierten Bilder nach diesen Anpassungen und erneutem Training bereits sehr nah an das gewünschte Ergebnis herankamen, wurde das Modell nun jeweils mit leicht angepassten Parametern für einen längeren Zeitraum von ungefähr zwei bis drei Tagen trainiert. Danach konnten die unterschiedlichen Modelle erneut evaluiert werden, um sich das beste Ergebnisse auszusuchen. Wie zu sehen ist war dieses Modell nun vollständig in der Lage den Artefakt freien Bereich des Bildes wiederzugeben, und die vom Artefakt verdeckten Bereiche des Gesichts überzeugend zu rekonstruieren. Die rekonstruierten Bereiche sind zwar teilweise in leicht geringerer Auflösung, aber insgesamt sind diese Fehler kaum bemerkbar.

Links: Originalbild Mitte: künstliches Artefakt Rechts: rekonstruiert

Wichtig hier ist zu beachten, dass das Modell weiterhin nur den wahrscheinlichsten Kontext einfüllt. Der Generator füllt die fehlende Information basierend darauf auf, was den Diskriminator im Training täuschen konnte. Da der Diskriminator hierbei mit den

echten Bildern vergleichen konnte, basiert der Bildinhalt also aus den Bildzusammenhängen der Trainingsdaten. Die Frau im folgenden Bild trug beispielsweise ursprünglich Schmuck auf ihrer Stirn. Dieses Objekt wird nun vollkommen vom Artefakt bedeckt, wodurch die Information vollkommen verloren geht. Da die Personen im Trainingsdatensatz keine Dekoration auf ihrer Stirn hatten, welches der Generator im Training generieren musste, wird das Objekt auch im rekonstruierten Input nicht abgebildet.

Original

Artefakt

rekonstruiert

Das Modell kann somit theoretisch auch verwendet werden, um kleinere Objekte gezielt zu entfernen, indem man diese absichtlich verdeckt, vorausgesetzt diese kamen nicht im Trainingsdatensatz vor.

Insgesamt lässt sich festhalten, dass die Verwendung von Generative Adversarial Networks für die Bildrekonstruktion ein vielversprechender Ansatz ist der bedeutende Fortschritte erzielt hat. Die Ergebnisse dieser Technik haben gezeigt, dass GANs das Potenzial haben in verschiedenen Bereichen wie medizinischer Bildgebung, Videoanalyse und digitaler Kunst eine revolutionäre Rolle zu spielen. Dennoch bleibt die Bildrekonstruktion mit GANs ein aktives Forschungsgebiet, welches auch in Zukunft noch weitere Herausforderungen und Verbesserungen erfordert. Zukünftige Arbeiten könnten sich hier vor allem auf eine höhere Auflösung der Bilder sowie auf eine größere Variation von Artefakten und Bildern fokussieren. Verbesserungen genannter Aspekte würde die Anwendbarkeit dieser Techniken im alltäglichen Leben drastisch steigern und somit auch das Öffentliche Interesse und Verständnis für künstliche Intelligenz wecken.

Dieses Projekt ist Open-Source und auf Github unter folgendem Link zu finden:
https://github.com/QntQ/Image-Reconstruction-using-GANs-and-Autoencoders

Quellenverzeichnis:

1. https://towardsdatascience.com/a-high-level-guide-to-autoencoders-b103ccd45924

2. https://www.tensorflow.org/datasets/catalog/mnist

3. https://www.tensorflow.org/datasets/catalog/fashion_mnist

4. https://towardsdatascience.com/denoising-autoencoders-dae-how-to-use-neural-networks-to-clean-up-your-data-cd9c19bc6915

5. https://github.com/NVlabs/ffhq-dataset

6. https://developers.google.com/machine-learning/gan/gan_structure

7. https://github.com/RaphaelMeudec/deblur-gan, Keras Implementierung von https://arxiv.org/pdf/1711.07064.pdf