Berliner Hochschule für Technik Berlin
Fachbereich VI – Informatik und Medien

Bachelorarbeit

Erzeugung von Bildern mittels Neuronalen Netzen

Stefan Berger
Medieninformatik
Matrikel-Nr. 854184

Berlin, 6. April 2021

Betreut von: Prof. Dr. F. Gers

Gutachter: Prof. Dr. J. Schimkat

Zusammenfassung

Im Experiment und im Inhalt dieser Bachelorarbeit werden die Fragen
1. ...
2. ...
...
beantwortet.

Inhaltsverzeichnis

1 Einleitung

Obwohl die Idee für eine Maschine, die anhand eingegebener Daten selbständig Entscheidungen treffen kann und die ersten praktischen Ansätze für künstliche neuronale Netze schon einige Jahrzehnte alt sind, findet der Einsatz derartiger Algorithmen erst seit einigen Jahren statt. Viele Erfindungen, die vor 30 bis 50 Jahren in Filmen und Serien Science Fiction darstellten, sind inzwischen nicht nur Realität, sondern auch alltagstauglich. Zu den wichtigsten Beispielen zählen verbale Schnittstellen an Computersystemen und auch Armbanduhren, autonome Fahrzeuge etwa in Gestalt von Parkassistenten und verschiedene Verfahren zur biometrischen Identitätsprüfung.

Andere rasche technologische Fortschritte aus der jüngeren Vergangenheit haben nicht immer nur die Lebensqualität der beteiligten Personen erhöht, sondern stellten durch Missbrauch gelegentlich sogar Gefahren dar. So wie das Internet auch zur Verbreitung von Falschinformationen und die sichere Verschlüsselung von gespeicherten Daten auch für Erpressungen genutzt werden kann, ist die Generierung von täuschend echten Bildern unter Umständen geeignet, persönlichen, finanziellen oder anders gearteten Schaden zu verursachen.

Weiterhin existieren bei der Auswahl der Trainingsdaten für künstliche neuronale Netze rechtliche Grenzen. Bilddaten sind in mehr als ausreichenden Mengen vorhanden, berücksichtigen aber zum Beispiel nicht immer das Recht am eigenen Bild. Für diese Bachelorarbeit sind die Anforderungen an die Bildqualität außerdem sehr hoch, da möglichst auch Texturen und Lichtreflexionen erlernt werden sollen. Modellgrafiken mit geeigneten Material- und Beleuchtungseigenschaften gibt es zwar auch, aber nur in weitaus geringeren Mengen. Für solche Fremdarbeiten, die sehr arbeitsaufwendig sind, wäre auch die Klärung der Nutzungsrechte erforderlich geworden. Für Trainingsergebnisse, die das mentale Modell einer breiten Nutzergemeinschaft reflektieren, sind grundsätzlich auch Daten aus möglichst vielen verschiedenen Quellen erforderlich.

Es ist deshalb eine Brücke geschlagen worden zwischen Trainingsdaten, die zum einen aus zufällig ausgewählte Benutzereingaben bestehen, und solchen, die bestimmte Qualitätseigenschaften erfüllen und in beliebiger Menge erstellt werden können. Ein künstliches neuronales Netz soll aus Skizzen des “Quick, Draw!”-Datasets von Google hochwertig gestaltete Figuren generieren. TODO: Hierfür kommen als Ein- und Ausgabedaten sowohl Bilddateien als auch die jeweils zugrundeliegenden Dateiformate NDJSON und Wavefront OBJ infrage.

1) Ziel der Arbeit

Die Arbeit verfolgt das Ziel, verschiedene bewährte Architekturen künstlicher neuronaler Netze zur Generierung von Bildern zu untersuchen und in einem praxisorientierten Zusammenhang zu testen. Ich zeige mehrere Möglichkeiten, wirklichkeitsnahe Bilder von Alltagsgegenständen aus Skizzen, die durch Benutzer erstellt wurden, mittels eines künstlichen neuronalen Netzes zu generieren. Die Bilddateien der Skizzen sowie die generierten Bilddatei-

en können dabei aus RBG-Pixelinformationen bestehen oder die Grafik mittels Bildkoordinaten beschreiben, wie es bei Vektorgrafiken und 3D-Modellen der Fall ist. Eine Anwendung der Ergebnisse ist beispielsweise als Feature eines Grafiktablets oder eines Smartboards denkbar.

2) Bisherige Arbeiten

Künstliche neuronale Netze finden erst seit wenigen Jahren breite Aufmerksamkeit, seit auch Heimcomputer in der Lage sind die hohe Anzahl der erforderlichen Rechenoperationen in annehmbarer Zeit auszuführen. Seitdem sind wenige, englischsprachige Einführungen in die Thematik entstanden. Ein häufig genanntes Buch ist “Deep Learning”, das online kostenfrei zugänglich ist [1]. Ebenfalls online kostenfrei ist das Buch “Dive into Deep Learning” [2]. Ein weiteres, praxisorientierteres Buch ist “Deep Learning with Python” [3], dessen zweite Auflage bald herausgegeben wird.

Als Architektur für ein künstliches neuronales Netz kommt für diese Arbeit zunächst derselbe Aufbau wie in “Image-To-Image-Translation” [4] zum Einsatz. Dieser setzt seinerseits auf Generative Adversarial Networks [5] und Conditional Generative Adversarial Networks [6] auf.

In “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks” [7] werden ebenfalls realitätsnahe Ergebnisse erzielt. Anders als bei den bisher erwähnten Arbeiten wird Unsupervised Learning verwendet, um das künstliche neuronale Netz zu trainieren.

2 Modell

1) Logistische Regression

2) Deep Neural Networks

3) Convolutional Neural Networks

Für Bilddaten wendet ein CNN diese Operation typischerweise auf zwei dreidimensionale Matrizen an, nämlich einerseits auf die Eingabedaten und andererseits auf einen sogenannten Filter oder auch Kernel. Die Eingabedaten sind in der ersten Schicht des Netzes die RGB-Pixelinformationen und in allen weiteren konvolutionalen Schichten die Ausgabe der vorherigen Schicht. Ein Filter ist eine Anzahl von trainierbaren Parametern, in diesem Fall auch eine . Beide Matrizen haben also die Form ($H\ x\ W\ x\ C$), wobei H die Höhe, W die Breite und C die RGB-Farbwerte repräsentiert.

Jede der drei Matrixdimensionen variiert üblicherweise zwischen den verschiedenen Schichten des Netzes. In fast allen CNNs ([1], [8], [9], [4]) nimmt die Kardinalität zunächst ab. Die Reduktion kann durch die Konvolution selbst entstehen oder durch Pooling-Schichten. Beim Pooling werden aus benachbarten Matrixkoeffizienten meist das Maximum, seltener der Durchschnitt oder andere Aggregierungen gebildet. Auf diese Weise wird das neuronale Netz darauf trainiert die relevanten Informationen zu extrahieren. [1]

Den konvolutionalen Schichten folgt in einigen Anwendungsfällen eine voll vernetzte Schicht (engl. Fully Connected Layer, FC), in der für jedes Neuron mit jedem Neuron der vorherigen Schicht eine Verbindung besteht. Besonders für die Bilderkennung ist diese Architektur gut geeignet. Die Ausgabe des neuronalen Netzes ist dann ein Vektor, beispielsweise von Wahrscheinlichkeitswerten für das Vorhandensein bestimmter Objekte und gegebenenfalls Bildkoordinaten der erkannten Objekte. Im Fall der Bildgenerierung ist die Ausgabe aber wieder eine Matrix von RGB-Pixelinformationen in der Form ($H\ x\ W\ x\ 3$).

4) U-Net-Architektur

Die bis hierhin beschriebenen neuronalen Netze besitzen eine gradlinige Struktur, in der die Ausgabe einer Schicht nur an die nächste Schicht übergeben wird. Bei zunehmender Anzahl der Schichten verbessert sich die Performance neuronaler Netze mit diesem Aufbau zunächst, aber verschlechtert sich bei zu vielen Schichten wieder. In einem Residual Neural Network (ResNet) verhindern zusätzliche Verbindungen zwischen nicht direkt aufeinanderfolgenden Schichten diesen Performanceverlust.

Ein U-Net ist eine spezielle Form eines ResNets. Es hat eine annähernd symmetrische Struktur, in der sich die Kardinalitäten der Matrizen zuerst verringern und anschließend

wieder erhöhen. U-Nets erzielen selbst mit wenigen Trainingsdaten gute Ergebnisse und benötigen dafür vergleichsweise wenig Rechenleistung. [10]

5) Generative Adversarial Networks

Ein Generative Adversarial Network besteht zunächst aus einem Generator und einem Discriminator [5]. Der Generator lernt während des Trainings täuschend echt aussehende Bilddaten zu generieren. Der Discriminator wird dagegen darauf trainiert, echte von generierte Bildern zu unterscheiden. Anschließend können beide Modelle "gegeneinander antreten". Deswegen wird es Generative *Adversarial* Network genannt.

6) Image-To-Image-Translation

3 Implementierung

1) Entwicklungsumgebung

1).1 Ubuntu Linux

1).2 Python

1).3 Tensorflow

1).4 CUDA

4 Experimente und Resultate

1) Vorbereitung der Eingabedaten

Die Trainingsdaten liegen im NDJSON-Format und als Blender-Dateien beziehungsweise Wavefront OBJ-Dateien vor. Die effizienteste Möglichkeit, die Skizzen und 3D-Modelle für die Verarbeitung in einem CNN vorzubereiten, ist das Rendern und Speichern als Bilddateien. Als Dateiformat kommen JPEG oder PNG infrage. Beide Formate können leicht als Trainingsset mit Tensorflow geladen werden.

Bei der Verarbeitung in einem Convolutional Neural Network spielt die Bildgröße in Bezug auf die Verarbeitungszeit eine wichtige Rolle. Bildformate der Größe 256x256 oder kleiner sind üblich und gut geeignet. Für ein GAN ist es zwar nicht erforderlich, aber sinnvoll, für Ein- und Ausgabedaten dieselben Dimensionen festzulegen. Bei der Bildgenerierung aus Skizzen unterscheiden sich Ein- und Ausgabdedaten natürlich bei die Farbtiefe. Während die Skizzen Graustufenbilder sind, besitzen die generierten Bilder drei Farbkanäle (RGB).

Sowohl während der Entwicklung als auch zur Laufzeit kann es vorteilhaft sein, Farbwerte statt auf der oft verwendeten Skala von 0 bis 255 als Fließkommazahlen im Bereich 0,0 bis 1,0 darzustellen. Auch für die Ausgaben der einzelnen Schichten eines künstlichen neuronalen Netzes kann diese Normalisierung durchgeführt werden (TODO: BatchNorm).

2) Anwendung herkömmlicher Shader

3) Hyperparameter

4) Performancebeobachtungen

5 Diskussion

Literatur

1. GOODFELLOW, I. u. a.: *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.
2. ZHANG, A. u. a.: *Dive into Deep Learning*. 2020. https://d2l.ai.
3. CHOLLET, F.: *Deep Learning with Python*. 1st. USA: Manning Publications Co., 2017. ISBN 1617294438.
4. ISOLA, P. u. a.: *Image-to-Image Translation with Conditional Adversarial Networks*. 2018. Abgerufen unter arXiv: 1611.07004 [cs.CV].
5. GOODFELLOW, I. J. u. a.: *Generative Adversarial Networks*. 2014. Abgerufen unter arXiv: 1406.2661 [stat.ML].
6. MIRZA, M.; OSINDERO, S.: *Conditional Generative Adversarial Nets*. 2014. Abgerufen unter arXiv: 1411.1784 [cs.LG].
7. RADFORD, A. u. a.: *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2016. Abgerufen unter arXiv: 1511.06434 [cs.LG].
8. LECUN, Y. u. a.: Object Recognition with Gradient-Based Learning. In: *Contour and Grouping in Computer Vision*. Springer, 1999.
9. RONNEBERGER, O. u. a.: U-Net: Convolutional Networks for Biomedical Image Segmentation. In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer, 2015, Bd. 9351, S. 234–241. LNCS. Auch verfügbar unter: http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a. (available on arXiv:1505.04597 [cs.CV]).
10. HE, K. u. a.: *Deep Residual Learning for Image Recognition*. 2015. Abgerufen unter arXiv: 1512.03385 [cs.CV].

Bildnachweis

Abkürzungs- und Symbolverzeichnis

Abkürzungen

AC	Air Compressor, Luftverdichter
APH	Air Preheater, Luftvorwärmer
CC	Combustion Chamber, Brennkammer
EXP	Expander
HRSG	Heat Recovery Steam Generator, Abhitzekessel

Lateinische Symbole

c	Spezifische Kosten je Exergieeinheit, €/J_{ex}
$\dot{C}$	Kostenstrom, €/h
CC	Kapitalgebundene Kosten, €
cf	Capacity Factor, Jährliche Auslastung, –
e	Spezifische Exergie, J/kg
$\bar{e}$	Spezifisch molare Exergie, J/mol
$\dot{E}$	Exergiestrom, W
f	Exergoökonomischer Faktor, –
fc	Spezifische Brennstoffkosten, €/J
FC	Brennstoffkosten, €
h	Spezifische Enthalpie, J/kg
$\dot{H}$	Enthalpiestrom, W
HHV	Brennwert, J/kg

Griechische Symbole

Δ	Differenz
ε	Exergetischer Wirkungsgrad, –
η_s	Isentroper Wirkungsgrad, –
κ	Isentropenexponenten, –
λ	Luftzahl, –

Hoch- und tiefgestelle Indizes

0	Referenzpunkt, Thermodynamische Umgebung
a	Avarage, Mittlere
D	Destruction, Vernichtung
F	Fuel, Brennstoff, Aufwand
net	Netto

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang