Was macht eine Schraube zu einer Schraube?

Storyboard

In der Technikum Digital Factory stellen wir zurzeit Roboter für Forschung- und Ausbildungszwecke her. Die Maschinen in der Fertigungsstraße werden dabei automatisch mit belieferten Komponenten wie z.B. Schrauben bestückt, damit die Produktion vollautomatisch abläuft. Durch diesen hohen Grad der Automatisierung fällt es aber oftmals erst spät auf, wenn fehlerhafte oder beschädigte Komponenten geliefert wurden. Beschädigungen fallen oft erst dadurch auf, dass das Teil nicht eingebaut werden kann; wodurch die ganze Maschine zum Stehen kommen kann. Also liegt es nahe, Schrauben und andere Teile vor dem Bestücken der Maschine zu kontrollieren. Da die händische Kontrolle bei großen Stückzahlen sehr zeitaufwändig ist, wollen wir diese automatisiert durchführen.

Aber wie erkennen wir automatisch Beschädigungen an gelieferten Teilen?

Um diese Frage zu beantworten, wollen wir mittels eines Kamerabilds Schrauben auf Beschädigungen überprüfen, bevor sie in eine Maschine eingelegt werden. Da die Produktion aber noch nicht in vollem Gange ist, haben wir nur einige wenige beschädigte Schrauben als Referenz zur Verfügung. Also können wir nur wenige Beispielbilder von beschädigten Schrauben aufnehmen, um unser System zu trainieren.

Wir benötigen also ein System welches lernt, visuell beschädigte von unbeschädigten Schrauben zu unterscheiden. Aber was macht eine Schraube überhaupt zu einer beschädigten Schraube?

Generative Adversarial Networks

In den AIAV Videos Classifier vs. Detector, CNN Classifier, Multinomial Naive Bayes Classifier und Gaussian Naive Bayes Classifier haben wir uns bereits verschiedene Arten der Klassifizierung angesehen. Dabei lernen Modelle bestimmte Aufgaben, z.B. das Erkennen eines Hammers, mithilfe von Trainingsdaten.

Dabei stellt sich die Frage, wo diese Daten für die Trainingssequenzen herkommen. Will man z.B. einen Hammer erkennen, braucht man mehrere Bilder von dem Hammer, den man erkennen will. Sollen jetzt mehrere verschiedene Arten von Hammern erkannt werden, sind mehrere Bilder von allen zu erkennenden Hammerarten notwendig. Die Anforderungen an qualitative Trainingsdaten werden schnell sehr hoch; je genereller die Klassifizierung, desto mehr Trainingsdaten braucht man.

Dieses Problem kann man mit Generative Adversarial Networks (GAN) umgehen. Die Idee von GANs ist es, zwei neuronale Netzwerke (siehe AIAV Video Klassische Neuronale Netze) gegeinander antreten zu lassen. Diese beiden Netzwerke heißen Generator und Diskriminator. Die Aufgabe des Generators ist es, aus Rauschen Daten zu generieren, welche zu einem vorgegebenen Datensatz passen. Der Diskriminator versucht dann, die echten von den generierten Daten zu unterscheiden.

Abbildung 1 zeigt echte sowie generierte Bilder von Schrauben. Die echten Bilder sind aus einem offenen Datensatz, während die generierten Bilder von einem, anhand diesen Datensatzes trainierten, GAN erzeugt wurden. Da das GAN anhand des Datensatzes trainiert wurde, erzeugt sein Generator

Bilder, welche zu den bestehenden Bildern im Datensatz passen. Die echten Bilder wurden dabei auf die gleiche Auflösung (32x32 Pixel) skaliert, wie die Bilder aus dem GAN. Können Sie erkennen, welche Bilder echt sind und welche nicht?

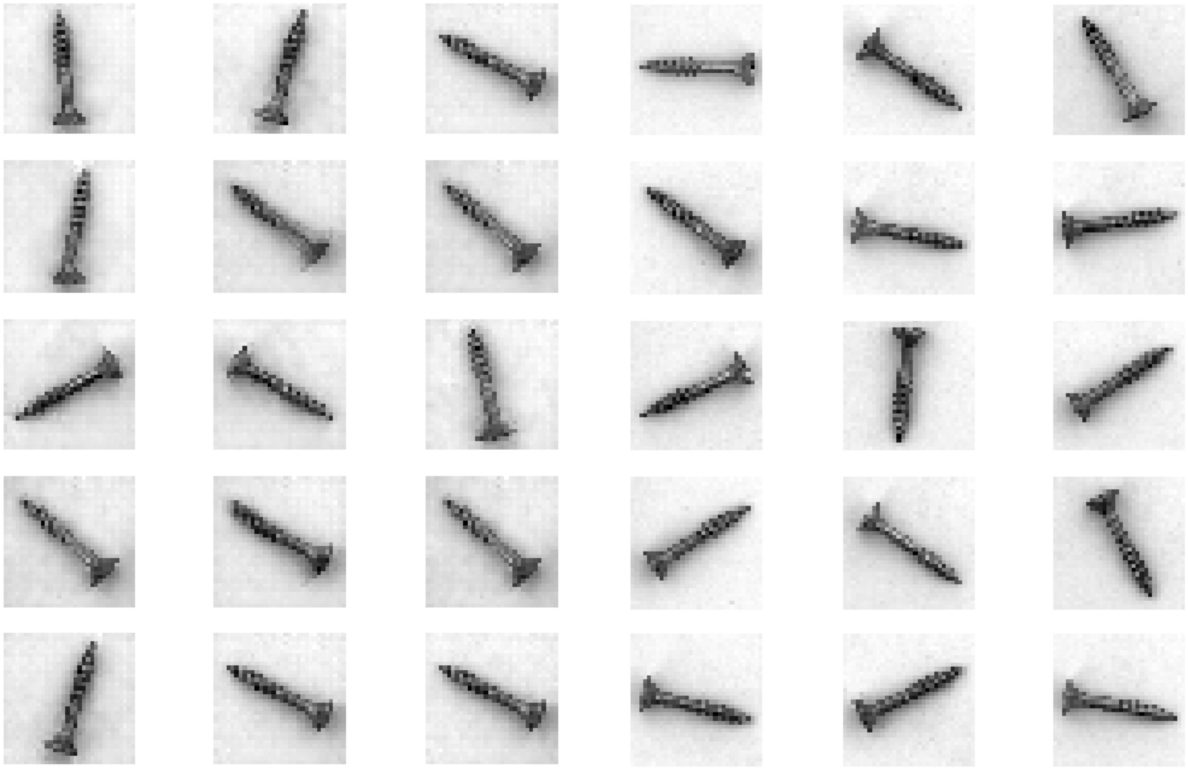

Abbildung 1: Hier sehen Sie echte sowie generierte Bilder von Schrauben. Die echten Bilder sind aus einem offenen Datensatz, während die generierten Bilder von einem, anhand diesen Datensatzes, trainierten Generative Adversarial Networks kommen. Können Sie erkennen, welche Bilder echt sind und welche nicht?

Abbildung 2 zeigt den Ablauf, wie GANs trainiert werden. Zunächst erzeugt der Generator auf Basis eines rauschenden Signals eine Probe. Diese Probe entspricht einem Eintrag im Trainingsdatensatz. Danach wird eine weitere Probe, dieses Mal aus dem Trainingsdatensatz, genommen. Nun werden beide Proben in den Diskriminator gegeben, welcher für jede der Proben schätzt, ob diese echt, also aus dem Trainingsdatensatz ist, oder vom Generator erzeugt wurde. Während des Trainings wird dieser Ablauf mehrmals durchgeführt, mit dem Ziel, gleichzeitig Generator und Diskriminator zu optimieren.

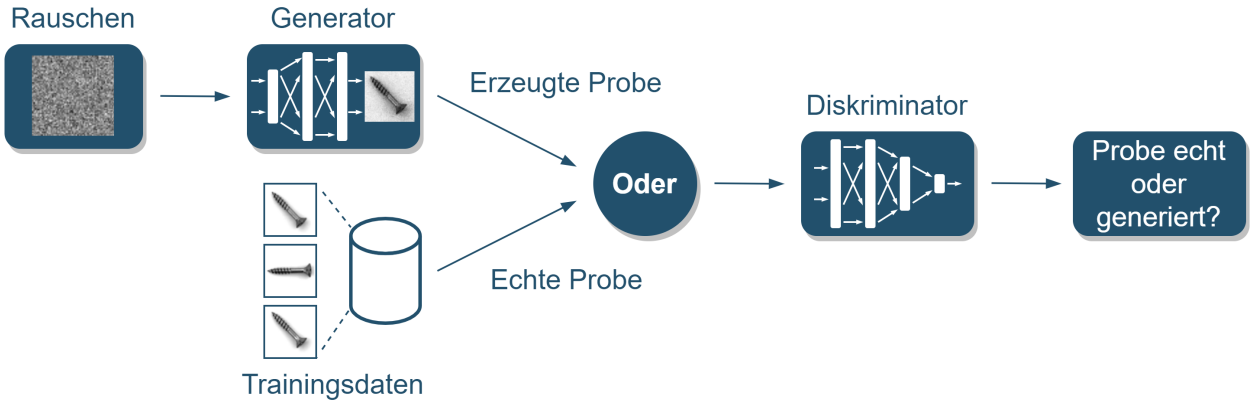

Abbildung 2: Die Idee von Generative Adversarial Networks (GAN) ist es, gleichzeitig den Generator und einen Diskriminator zu trainieren. Der Generator versucht dabei aus Rauschen einen Ausgang zu erzeugen, welcher der Probe aus den Trainingsdaten ähnlich ist. Diese generierten Datenproben

werden dann zusammen mit Proben aus dem Trainingsdatensatz dem Diskriminator übergeben. Die Aufgabe des Diskriminators ist dann zu unterscheiden, welche Proben echt und welche generiert sind. Der Ablauf von GAN ist hier mittels dem Datensatz von Schraubenbildern gezeigt.

Nach dieser Trainingssequenz können wir nun Generator und Diskriminator speichern und für andere Aufgaben verwenden. In unserem Beispiel kann der Generator nach genug Trainingsepisoden Bilder von Schrauben erzeugen, welche nicht von den Trainingsbildern unterscheidbar sind. Der Diskriminator hat währenddessen gelernt zu erkennen, ob das eingegebene Bild von einer Schraube ist, oder nicht.

Praktische Implementierung

Für die praktische Implementierung des Usecases trainieren wir ein GAN anhand von aufgenommenen Bildern von unbeschädigten Schrauben. Dabei werden die Bilder zunächst als Graustufenbild eingelesen und auf eine Auflösung von 32 mal 32 Pixel skaliert. Die Skalierung auf so eine niedrige Auflösung ist notwendig, um den Rechenaufwand des Trainings gering zu halten. Nach dem Training werden Generator und Diskriminator zur Klassifizierung wiederverwendet. Dabei wird ein Modell zur Erkennung von Anomalien basierend auf Generator und Diskriminator trainiert. Dieses entscheidet, ob neue und damit unbekannte Bilder die gleichen Merkmale aufweisen, wie die Bilder im Trainingsdatensatz.

Abbildung 3 zeigt den Ablauf der Lösung. Dabei führen wir die Trainingssequenz vorab anhand der Trainingsdaten durch und speichern die Trainierten Netzwerke ab, da der benötigte Rechenaufwand für das Training relativ hoch ist. Während des Betriebs lesen wir dann Schraubenbilder ein, bearbeiten sie genauso wie die Trainingsdaten vor und lassen das Modell entscheiden ob das Bild Anomalien aufweist. Anomalien sind in diesem Fall Beschädigungen an der Schraube.

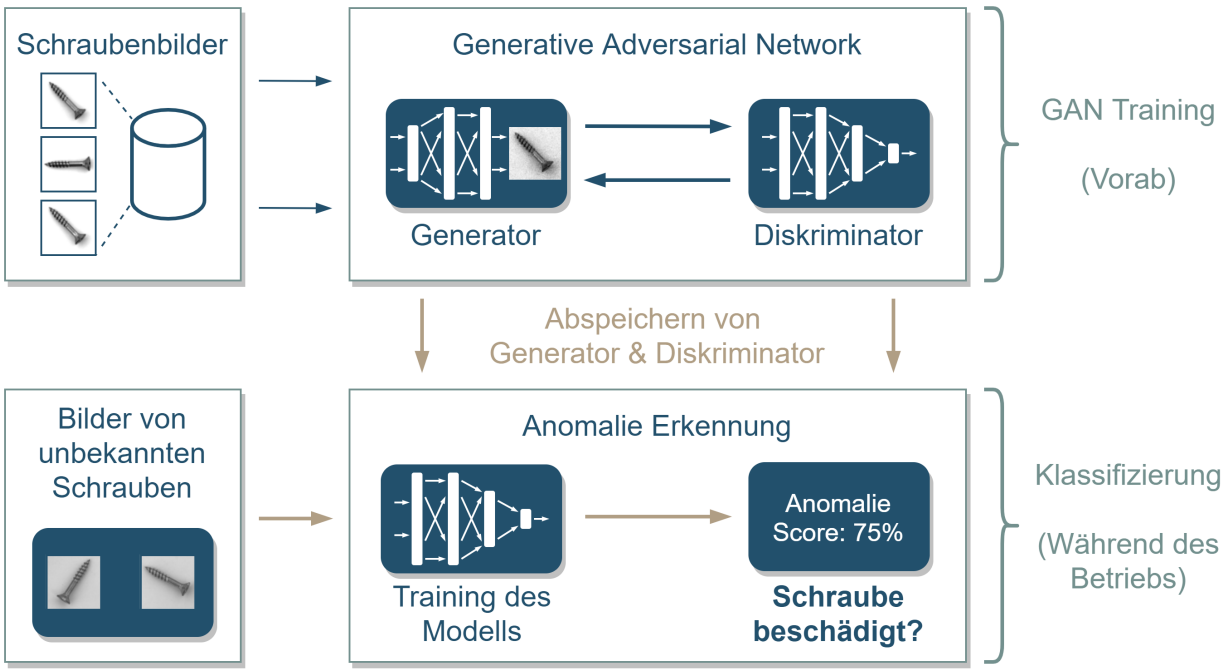

Abbildung 3: Wir verwenden Generator und Diskriminator aus dem GAN wieder, um zu erkenen ob unsere Schrauben beschädigt sind. Die beschädigten Schrauben passen sie nicht zu den Schrauben aus dem Trainingsdatensatz und werden daher als beschädigt erkannt.

Fazit

Generative Adversarial Networks erlauben es uns zu lernen, auch wenn nicht genug Daten für das Training von anderen Arten von Modellen vorhanden sind. Obwohl wir mit kleinen Neuronalen Netzwerken und in einer niedrigen Auflösung von 32x32 Pixeln gearbeitet haben, war die benötigte Rechenleistung für das Training relativ hoch. Dafür sind GANs aber eine sehr flexible Lösung. Vor allem in Kombination mit anderen Verfahren aus AI erlauben sie es uns, praktische Probleme zu lösen, ohne zuerst eine große Menge an Trainingsdaten aufnehmen zu müssen.