Einführung Computer Vision

Computer Vision ist ein interdisziplinäres Feld, das sich mit dem automatisierten Verstehen und Interpretieren von visuellen Informationen durch Computer befasst. Es kombiniert Techniken aus der Informatik, Mathematik, und Künstlicher Intelligenz, um Maschinen das Sehen beizubringen.

Dabei unterscheidet man unter anderem zwischen: - Erkennung: Identifizieren und Klassifizieren von Objekten in Bildern oder Videos. - Segmentierung: Aufteilen eines Bildes in seine Bestandteile und Zuweisen von Bedeutungen zu diesen Segmenten. - Bewegungsverfolgung: Verfolgen von Objekten über mehrere Bilder oder Videoframes hinweg. - Rekonstruktion: Wiederherstellung der 3D-Struktur einer Szene aus 2D-Bildern.

Anwendungsbereiche: - Medizinische Bildgebung: Automatisierte Analyse von Röntgenbildern und MRT-Scans. - Autonome Fahrzeuge: Erkennung von Straßen, Verkehrszeichen und anderen Fahrzeugen. - Überwachung: Automatische Erkennung von verdächtigem Verhalten in Sicherheitskameras. - Industrieautomation: Qualitätskontrolle und Fehlererkennung in Produktionslinien.

--> [] Überlegen Sie sich 2 weitere Beispiele in denen Computer Vision eingesetzt werden könnte.

Einführung in Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) sind spezielle Arten von neuronalen Netzwerken, die besonders gut für die Verarbeitung von Bildern und anderen grid-ähnlichen Datenstrukturen geeignet sind. Sie haben sich als äußerst effektiv in der Bild- und Videoerkennung, Bildklassifikation und vielen anderen Aufgaben der Computer Vision erwiesen.

Struktur und Funktionsweise von CNNs

Convolution Layer (Konvolutional-Schicht): - Funktionsweise: Ein Convolution Layer wendet mehrere Filter (auch Kernel genannt) auf das Eingabebild an. Diese Filter

verschieben sich über das Bild und berechnen dabei das Skalarprodukt zwischen dem Filter und den verschiedenen Teilen des Bildes. - Filter/Kernel: Ein Filter ist eine kleine Matrix, die über das Bild gleitet und Merkmale wie Kanten, Texturen oder Muster erkennt. Die Größe und Anzahl der Filter beeinflussen, welche Merkmale erkannt werden. - Ergebnis: Das Ergebnis der Konvolution ist eine Feature Map, die die Position und Stärke der erkannten Merkmale zeigt.

Pooling Layer (Pooling-Schicht): - Funktionsweise: Ein Pooling Layer reduziert die räumliche Größe der Feature Maps, um die Berechnung zu beschleunigen und die Robustheit gegenüber kleinen Verschiebungen und Verzerrungen zu erhöhen. - Arten von Pooling: - Max Pooling: Wählt das maximale Element aus jeder Region der Feature Map. - Average Pooling: Berechnet den Durchschnittswert jeder Region der Feature Map. - Ergebnis: Eine verkleinerte Version der Feature Map, die wichtige Merkmale beibehält und unwichtige Details entfernt.

Fully Connected Layer (Vollständig verbundene Schicht): - Funktionsweise: In einer Fully Connected Layer ist jeder Neuron mit jedem Neuron der vorherigen Schicht verbunden. Diese Schicht kombiniert die extrahierten Merkmale und führt die eigentliche Klassifikation durch. - Bedeutung: Die Fully Connected Layer ermöglicht es dem Netzwerk, nichtlineare Kombinationen der extrahierten Merkmale zu lernen und komplexe Muster in den Daten zu erkennen.

Beispielhafte Struktur eines CNNs: - Eingabeschicht: Akzeptiert das Eingabebild. - Konvolutional-Schicht: Extrahiert lokale Merkmale. - ReLU (Rectified Linear Unit): Führt eine nichtlineare Aktivierungsfunktion durch. - Pooling-Schicht: Reduziert die Größe der Feature Maps. - Mehrere Konvolutional- und Pooling-Schichten: Weiteres Extrahieren und Verdichten von Merkmalen. - Flatten Layer: Wandelt die 2D-Feature Maps in einen 1D-Feature Vektor um. - Fully Connected Layer: Klassifiziert die extrahierten Merkmale. - Ausgabeschicht: Gibt die Wahrscheinlichkeiten für jede Klasse aus.

```
import tensorflow as tf
from tensorflow.keras import layers, models
import numpy as np

# Beispiel-Daten (z.B. MNIST Dataset)
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()

# Vorverarbeitung der Daten
# ...

# Aufbau des CNN-Modells
model = models.Sequential()

```

```
# Eingabeschicht (in diesem Fall implizit durch die erste Konvolutionsschich
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28,
model.add(layers.MaxPooling2D((2, 2)))

# Weitere Konvolutions- und Pooling-Schichten
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))

# Flatten Layer
model.add(layers.Flatten())

# Fully Connected Layer
model.add(layers.Dense(64, activation='relu'))

# Ausgabeschicht
model.add(layers.Dense(10, activation='softmax'))

# Kompilieren des Modells
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Training des Modells
model.fit(x_train, y_train, epochs=5, validation_data=(x_test, y_test))

# Evaluierung des Modells
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print(f'\nTest accuracy: {test_acc}')
```

Erklärung des Codes:

1. Convolutional Layer (Konvolutionsschicht)

```
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1
```

Parameter: - 32: Die Anzahl der Filter (auch "Kernels" genannt) in der Konvolutionsschicht. Jeder Filter lernt, ein unterschiedliches Merkmal des Eingabebildes zu erkennen, wie z.B. Kanten, Ecken oder Muster. In diesem Fall verwendet die Schicht 32 Filter. - (3, 3): Die Größe der Filter ist 3x3. Das bedeutet, dass jeder Filter ein 3x3-

Pixel-Feld des Eingabebildes abtastet. Kleinere Filter wie 3x3 sind häufig in CNNs, da sie eine gute Balance zwischen Erkennungsfähigkeit und Rechenaufwand bieten. - activation='relu': Die Aktivierungsfunktion, die nach der Konvolution angewendet wird. ReLU (Rectified Linear Unit) ist eine weit verbreitete Aktivierungsfunktion in CNNs, weil sie die nichtlineare Transformation einführt und gleichzeitig das Vanishing-Gradient-Problem reduziert. - input_shape=(28, 28, 1): Die Form der Eingabedaten. In diesem Fall handelt es sich um 28x28-Pixel-Bilder mit 1 Kanal (graustufige Bilder). Dieser Parameter ist nur in der ersten Schicht erforderlich, um die Form der Eingabedaten anzugeben.

1. MaxPooling Layer(Pooling-Schicht)

```
model.add(layers.MaxPooling2D((2, 2)))
```

Parameter: - (2, 2): Die Größe des Pooling-Fensters ist 2x2. Das bedeutet, dass die Pooling-Schicht auf nicht überlappende 2x2-Blöcke in der Eingabe angewendet wird und das Maximum jedes Blocks extrahiert wird. Ein 2x2 Pooling-Fenster ist weit verbreitet, weil es die räumlichen Dimensionen der Feature Maps effizient um die Hälfte reduziert. Dies hilft, die Anzahl der Parameter und die Rechenlast zu verringern, was zu einem schnelleren und effizienteren Training führt. MaxPooling wählt das maximale Element in jedem 2x2-Fenster, was dazu beiträgt, dominante Merkmale beizubehalten und unwichtige Details zu eliminieren.

Bildklassifikation vs. Objekterkennung vs. Segmentation

Bildklassifikation

Die Bildklassifikation umfasst die Identifizierung, welche Objekte in einem Bild vorhanden sind. Das Hauptziel ist es, ein Label oder eine Klasse einem gesamten Bild zuzuordnen. Zum Beispiel könnte ein Klassifizierungsmodell anhand eines Bildes bestimmen, dass es eine „Katze“ oder einen „Hund“ enthält.

Eingabe: Ein Bild. Ausgabe: Ein einzelnes Label, das das wahrscheinlichste Objekt im Bild repräsentiert. Anwendungsfall: Identifizierung des Hauptmotivs eines Bildes, wie die Unterscheidung zwischen verschiedenen Arten von Tieren, Fahrzeugen oder Produkten.

Objektdetektion

Die Objektdetektion geht einen Schritt weiter als die Klassifizierung, indem sie nicht nur identifiziert, welche Objekte in einem Bild vorhanden sind, sondern auch deren Position bestimmt. Dies umfasst das Zeichnen von Begrenzungsrahmen (Bounding Box) um jedes Objekt und die Klassifizierung dessen, was jedes Objekt ist.

Eingabe: Ein Bild. Ausgabe: Mehrere Labels und Bounding Boxen. Jeder Bounding Box entspricht einem Objekt, und jeder Rahmen hat ein Label und eine Konfidenzbewertung. Anwendungsfall: Anwendungen, die sowohl die Kenntnis der Anwesenheit als auch der Position von Objekten erfordern, wie autonomes Fahren, bei dem das System Fußgänger, andere Fahrzeuge und Hindernisse erkennen und lokalisieren muss.

Objektsegmentierung

Die Objektsegmentierung bietet eine detailliertere Analyse, indem die genauen Pixel identifiziert werden, die zu jedem Objekt gehören. Es gibt zwei Haupttypen der Segmentierung: Semantische Segmentierung und Instanzsegmentierung.

Semantische Segmentierung: Dies umfasst die Klassifizierung jedes Pixels in einem Bild in eine Kategorie. Sie unterscheidet jedoch nicht zwischen verschiedenen Instanzen derselben Klasse. Zum Beispiel, wenn es zwei Hunde in einem Bild gibt, wird die semantische Segmentierung alle Hundepixel als „Hund“ markieren, ohne zwischen den beiden Hunden zu unterscheiden. Eingabe: Ein Bild. Ausgabe: Eine Maske, bei der jeder Pixel mit einer Klasse gekennzeichnet ist. Anwendungsfall: Aufgaben, bei denen eine präzise pixelgenaue Klassifizierung erforderlich ist, wie in der medizinischen Bildgebung (Segmentierung von Tumoren) oder Hintergrundentfernung.

Instanzsegmentierung: Diese kombiniert die Objektdetektion und die semantische Segmentierung, indem sie nicht nur die Klasse jedes Pixels identifiziert, sondern auch zwischen verschiedenen Instanzen derselben Klasse unterscheidet. Zum Beispiel, wenn es zwei Hunde in einem Bild gibt, wird die Instanzsegmentierung jeden Hund separat kennzeichnen. Eingabe: Ein Bild. Ausgabe: Ein Satz von Masken, bei dem jede Maske einer anderen Instanz eines Objekts entspricht, zusammen mit Klassenlabels für jede Instanz. Anwendungsfall: Komplexe Aufgaben, die die Unterscheidung zwischen mehreren Instanzen von Objekten erfordern, wie in der Robotik (Aufnehmen bestimmter Gegenstände) oder fortschrittliche Bildbearbeitung.

Figure 1: Classification vs. Detection vs. Segmentation