Modul 4

Supervised Learning

"Das **Modell** *besteht aus* **Parametern**, *die nicht von Programmierer*innen selbst gewählt werden, sondern mithilfe von* **Training** *von einem* **Algorithmus angepasst** *werden ."*

Über das Modul

In diesem Modul geht es um **Supervised Learning** (***SL***). Ziel ist es, den Schüler*innen ein grundlegendes Verständnis zu vermitteln, was ***SL*** ist, was es kann und was nicht und wie man ***SL***-Algorithmen trainieren kann. Dieses Modul legt seinen Fokus auf die Praxis, daher werden die Schüler*innen ihre eigene Algorithmen trainieren und die Möglichkeiten und Probleme hautnah erleben. Ein vertiefender technischer Teil wird größtenteils übersprungen, da in anderen Modulen, wie beispielsweise jenem zu **Neuronalen Netzwerken** genauer darauf eingangen werden soll und für die praktische Arbeit ein Hintergrundwissen nicht notwendig ist.

Ziele

Die Schüler*innen sind in der Lage...

- ...***SL*** als eine Mapping Funktion zu erklären
- ...den Wert und die Anforderungen von Trainingsdaten zu benennen
- ...um abzuschätzen, ob eine Anwendung ***SL*** verwendet
- ...Probleme und Einschränkungen von ***SL*** zu nennen
- ...ihr eigenes ***SL***-Modell zu trainieren und es in einer Anwendung zu verwenden

Agenda

Zeit	Inhalt
30 min	**Einführung**
30 min	**Übung - Trainiere dein erstes Modell**
15 min	**Theorie - Vertrauen auf Daten**
30-60 min	**Übung - Game controller**
30 min	**Diskussion + Quiz - Möglichkeiten und Grenzen**

Einführung

In der Einführung geht es darum, dass sich die Schüler*innen mit den grundlegenden Konzepten und Begriffen vertraut machen, die beim **Supervised Learning** (***SL***) verwendet werden. Die vorgestellte Theorie basiert auf den **Folien**, die nicht nur allgemeine Anwendungen vorstellen, sondern auch die Vorstellung davon, was ein ***SL***-Algorithmus tut und wie ein ***SL***-Algorithmus trainiert werden kann.

Beispiele aus der realen Welt

(Folien 2 - 8)

In den ersten Folien werden als Beispiele mehrere Anwendungen für ***SL***-Algorithmen vorgestellt, die deutlich machen sollten, dass sie in vielen verschiedenen Anwendungen verwendet werden.

Typische Beispiele für ***SL*** sind das **Klassifizieren** von Bildern und das **Erkennen** von Objekten in Bildern. Die Gesichtserkennung kann beispielsweise verwendet werden, um eine Kamera automatisch auf Objekte zu fokussieren oder im Fall von Social-Media-Sites automatische Labels mit Namen zu erstellen. Das Ziel der Objekterkennung ist es, verschiedene Objekte innerhalb eines Bildes zu erkennen und zu identifizieren. Daher ist der erste Schritt normalerweise die Segmentierung, bei der das Bild in verschiedene Bereiche geclustert wird, die dann mit einem der verfügbaren Labels klassifiziert werden. All diese Schritte werden heutzutage meistens mit ***SL*** und **Deep Neural Networks (DNN)** durchgeführt. Auch in der Fertigungsindustrie werden ***SL*** und Bildklassifikation eingesetzt, um beispielsweise Defekte oder Montagefehler automatisch zu erkennen. Dies ist einer der Bereiche, in denen ***KI*** aufgrund geringerer Kosten und manchmal sogar aufgrund weniger Fehler immer mehr menschliche Arbeitskräfte ersetzt. In Bereichen wie der Medizin werden jedoch Algorithmen zur Erkennung von Krankheiten in Zusammenarbeit mit Ärzten verwendet, um ihnen zu helfen, die richtigen Schlüsse zu ziehen. Dies geschieht beispielsweise, indem auf Scans auf potenzielle Krebsherde hingewiesen wird, die sonst möglicherweise übersehen worden wären.

Das nächste Beispiel ist Spracherkennung und **Natural Language Processing (NLP)**. Anstelle eines Bildes werden Schallwellen verwendet, um zwischen gesprochenen Wörtern und anderen Geräuschen zu unterscheiden. Sobald Wörter erkannt worden sind, können sie transkribiert werden, was beispielsweise für die

automatische Untertitelgenerierung verwendet wird. Darüber hinaus können Sätze auf ihre Bedeutung analysiert werden, sodass Anwendungen wie digitale Assistenten (Alexa, Siri, Google Now, …) komplexe Befehle und Abfragen verstehen und sinnvoll beantworten/agieren können. Diese Art der Analyse von Sätzen/Texten wird auch in Spam-Filtern verwendet, um zwischen normaler Mail und Spam-Mail zu unterscheiden.

Eine weitere Verwendung von ***SL***-Algorithmen ist die Vorhersage von Werten wie den Kosten eines Hauses oder den Einnahmen eines geplanten Films, basierend auf verschiedenen Informationen wie der Größe des Hauses oder Schauspieler*innen im Film. In diesem Fall wird ein ***SL***-Algorithmus anhand von Daten über ähnliche Häuser und deren Kosten trainiert und dann verwendet, um den Wert neuer Häuser vorherzusagen.

Zu guter Letzt werden ***SL***-Algorithmen auch verwendet, um Benutzer*innen in verschiedene Profile einzuordnen, um so kundenspezifische Werbung bereitzustellen. Da die Anzeige eher den Interessen der Nutzer*innen entspricht, besteht eine höhere Chance, dass sie auf den Link klicken, was zu einer höheren Chance führt, das Produkt am Ende zu verkaufen.

Was Supervised Learning Algorithmen tatsächlich leisten

(Folien 9 - 13)

Nach all den Beispielen werden Supervised-Learning-Algorithmen als Mapping-Funktionen eingeführt. Der Hauptpunkt hier ist, dass es sich hierbei um eine sehr einfache und nachvollziehbare Sache handelt, bei der eine Eingabe eine Ausgabe erzeugt. Es gibt keine magische, undurchsichtige ***KI***-Entscheidungsfindung, es ist nur eine mathematische Formel, die ein klares Ergebnis liefert. Es ist eine gute Idee, sich die vorherigen Beispiele anzusehen und darüber zu sprechen, was die Eingabe und was die Ausgabe sein könnte. Wenn die Eingabe beispielsweise ein Bild ist, könnte die Ausgabe Positionen und Bezeichnungen verschiedener Objekte oder nur eine Bezeichnung wie „defekt“ oder „in Ordnung“ sein. Wenn die Eingabe ein Tonbeispiel ist, könnte die Ausgabe die entsprechenden Wörter sein, und wenn die Eingabe allgemeine Informationen über eine Filmproduktion sind, könnte die Ausgabe der erwartete Umsatz sein. Diese Arten von Problemen werden als **Klassifizierung** (Kennzeichnung von Daten) und **Regression** (Vorhersage von Werten) bezeichnet.

Es ist wichtig zu erwähnen, dass die Ausgabe normalerweise nicht binär ist, sondern schrittweise. Bei einem Bild einer Katze wäre der Algorithmus beispielsweise zu 99,5% sicher, dass es sich um eine Katze handelt, zu 0,4% sicher, dass es sich um einen Hund handelt, und zu 0,1% sicher, dass es sich um einen Fisch handelt.

Wie man Supervised Learning Algorithmen trainiert

(Folien 14 - 26)

Im letzten Abschnitt wird der Prozess des **Trainings** vorgestellt. Er basiert auf der Idee, dass Entwickler*innen den Algorithmus mit Beispielen versorgen (z. B. das ist ein Hund) und das Programm sein internes **Modell** (z. B. Variablen in einer Formel) anpasst, um das richtige Ergebniss auszugeben. Außerdem wird das Konzept der **Trainings-** und **Testdaten** eingeführt.

Es gibt dazu fünf wichtige Definitionen:

Supervised Learning Algorithmen

Ein **Algorithmus** ist eine schrittweise Beschreibung einer Aufgabe. In diesem Fall definiert er genau, wie der ***SL***-Prozess funktioniert, wie er lernt sowie welche **Parameter** verwendet werden und wie sie das Ergebnis beeinflussen. Es wird oft mit einem Kochrezept verglichen, wo es detaillierte Anweisungen gibt, die, wenn sie richtig befolgt werden, zu einem schmackhaften Ergebnis führen.

Parameter

Parameter sind Daten, die verwendet werden können, um das Ergebnis oder den Arbeitsprozess eines Algorithmus zu modifizieren. Um mit dem Kochbeispiel fortzufahren, könnte ein Parameter beim Backen eines Kuchens sein, wie süß er sein soll. Der Koch entscheidet dann über die Zuckermenge und folgt dem Rezept entsprechend. In ***SL*** werden Parameter im Trainingsprozess verwendet, bei dem die Programmierer*innen entscheiden müssen, auf welche Weise (wie viele Iterationen, interne Struktur, worauf er sich konzentrieren soll, ...) der Algorithmus lernen muss.

Supervised Learning Modell

Das Modell besteht aus Parametern, die nicht vom Programmierer, von der Programmiererin ausgewählt, sondern in einem Prozess namens **Training vom Algorithmus angepasst** werden. Im Kochbeispiel könnte dies eine Zutatenliste mit der entsprechenden Menge sein, die der Algorithmus dann

während des Trainings optimieren muss, um das gewünschte (z. B. schmackhafte) Ergebnis zu erhalten. Gerade in der Bildklassifikation werden oft **Neuronale Netzwerke** als zugrunde liegendes Modell verwendet, daher werden Modelle auch oft nur **Netzwerke** genannt.

Klassifizierte Daten

Ein Datensatz, der nicht nur die Eingabe, sondern auch die Ausgabe des ***SL***-Algorithmus enthält. Dies können beispielsweise einige hundert Bilder von Katzen und Hunden mit den entsprechenden Bezeichnungen „Katze“ und „Hund“ sein.

Trainings- und Test-Daten

Normalerweise beginnt man damit, viele **markierte Daten** zu sammeln, die dann in zwei Sätze aufgeteilt werden, die Trainingsdaten und die Testdaten. Die Trainingsdaten werden verwendet, um das Modell zu trainieren, während die Testdaten verwendet werden, um seine Genauigkeit zu testen. Es ist wichtig, dass es sich um **unterschiedliche** Daten handelt, um sicherzustellen, dass das Modell allgemein genug ist und nicht auf die Besonderheiten der Trainingsdaten fixiert ist.

Sobald diese Begriffe klar sind, ist der Prozess des Trainierens eines Modells ganz einfach:

1. Sammel einen Satz von **klassifizierten Daten** und teile ihn in zwei Sätze auf (Trainings- und Testdaten).
2. **Beginne dein Modell** mit beliebigen Werten (oft zufällig) und wähle einen Satz von **Trainingsparametern** aus
3. **Wiederhole** das Training mit den Trainingsdaten (so oft wie durch die Parameter definiert)
    - Anhand einer Eingabe **passe** die Werte innerhalb des Modells an, um sie an die gegebenen Ausgabe besser anzupassen
4. **Teste dein Modell** mithilfe von Testdaten, um seine Genauigkeit zu überprüfen
5. **Wiederhole den Prozess von Anfang an** und ändere die klassifizierten Daten oder Trainingsparameter, bis du mit deinem Modell zufrieden bist.

Fragen und was kommt als nächstes

(Folien 26)

Zu diesem Zeitpunkt sollten die Schüler*innen die Grundlagen der Funktionsweise von ***SL***-Algorithmen kennen. In den kommenden Übungen werden sie den zuvor

beschriebenen Prozess dann selbst ausprobieren und herausfinden, welche **Probleme** dabei auftreten können und wie sensibel Algorithmen auf ihre Trainingsdaten reagieren.

Obwohl dieser Kurs nicht auf technische Details eingeht, ist es dennoch erwähnenswert, dass es je nach Problem mehrere, unterschiedliche Algorithmen gibt, die für ***SL*** verwendet werden können. Die meisten Menschen haben von neuronalen Netzen gehört, die für viele Probleme verwendet werden, insbesondere wenn die Daten recht komplex sind, wie bei der Bild- oder Spracherkennung. Aber es gibt noch viele weitere Algorithmen wie Entscheidungsbäume, Regressionskurven, nearest neighbor und so weiter.[1]

Material

- SL - Introduction.pdf

Referenzen

1. https://builtin.com/data-science/supervised-machine-learning-classification

Trainiere dein erstes Modell

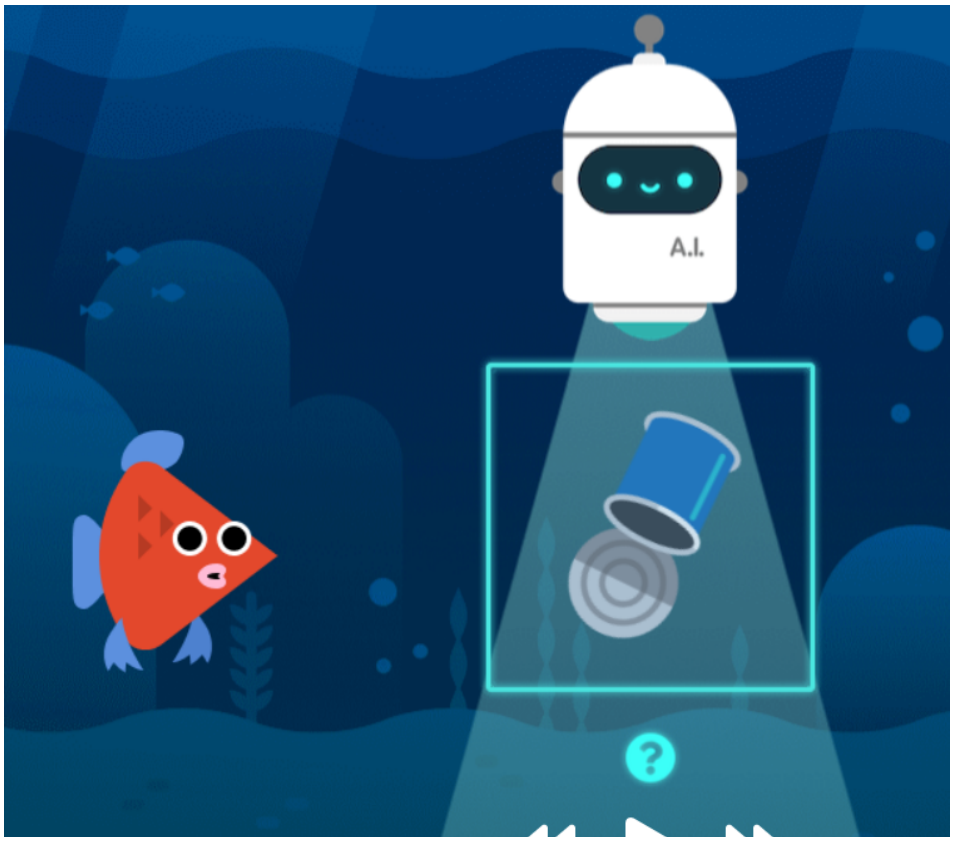
Ziel dieser Übung ist es, den Schüler*innen einen ersten Eindruck zu vermitteln, wie das **Training** eines ***SL***-Algorithmus funktioniert und welche **Schwierigkeiten** dabei auftreten können. Die wichtigste Erkenntnis sollte sein, dass das Training stark von den **Trainingsdaten** abhängt und dass es im Allgemeinen ein Prozess ist, der viele Wiederholungen benötigt, um zu einem nützlichen Ergebnis zu führen.

Es gibt zwei mögliche Übungen:
AI for Oceans ist eine Online-Übung und das **Klassifikations-Spiel** eine praktische Pen&Paper Übung.

AI for Oceans

Diese Übung arbeitet mit der Ocean-Scanning-Übung aus dem Kurs AI for Oceans. Mithilfe eines **Webbrowsers** müssen die Schüler*innen einen Roboter trainieren, um zwischen Fischen und Müll sowie zwischen verschiedenen Arten von Wassertieren zu unterscheiden.

Übung / AI for Oceans



Klassifikations-Spiel

Diese Übung versetzt die Schüler*innen in die Rolle eines ***SL***-Algorithmus, sodass sie die Schwierigkeiten beim Klassifizieren von Bildern von Katzen und Hunden in dieser **Pen and Paper**-Übung kennen lernen können.

Übung / Klassifikations-Spiel

Ziele

Die Schüler*innen sind in der Lage...

- ...den Prozess des Trainierens eines ***SL***-Algorithmus zu erklären
- ...zu erklären, wie Algorithmen funktionieren, indem man sich auf verschiedene Merkmale konzentriert
- ...zu erklären, wie sehr das System von seinen Trainingsdaten abhängig ist

Vertrauen auf Daten

In diesem Kapitel geht es um **Trainingsdaten** und den Einfluss und die Bedeutung der **Qualität** dieser Daten. Während sich die Beispiele in diesem Kapitel auf die Bilderkennung konzentrieren, gelten alle diskutierten Probleme dennoch für alle Formen des Supervised Learning. Das Kapitel baut auf **diesen Folien** auf.

Overfitting und Underfitting

(Folien 2-14)

Die beiden größten Probleme beim Trainieren eines ***SL***-Algorithmus sind **Overfitting** und **Underfitting**.

Bei **Overfitting** handelt es sich um das Problem, dass ein Modell zwar **während des Trainings** gut funktioniert, aber bei **anderen Daten** schlechte Ergebnisse liefert. Es tritt auf, wenn das Trainingsset entweder zu spezifisch ist oder sehr auffällige und irreführende Merkmale aufweist.

Underfitting hingegen bedeutet, dass das Modell überhaupt nichts Nützliches gelernt hat. Es tritt auf, wenn die Eingabedaten zahlenmäßig **unzureichend** oder **zu allgemein** sind und keine zugehörigen Merkmale gefunden werden können. Sehen wir uns einige Beispiele dazu an.

Stell dir vor, du möchtest ein Modell erstellen, das zwischen verschiedenen Personen unterscheidet, indem es Porträtbilder analysiert. Du bittest Freunde, dir ein paar hundert Porträtbilder zu geben, die mit ihrer Webcam aufgenommen wurden, und verwendest sie, um deinen Algorithmus zu trainieren. Obwohl die Testergebnisse sehr vielversprechend sind, stellst du fest, dass beim Testen deines Modells, bei dem du deine Freunde zu dir nach Hause einladest, kaum jemand richtig erkannt wird.

Dies ist ein klassisches Beispiel für **Overfitting**, bei dem das Modell unter bestimmten Umständen gut funktioniert, aber schrecklich versagt, wenn man die Umgebung ändert. Ein Grund für das Scheitern könnte sein, dass das Modell nicht wie erwartet die Merkmale von Gesichtern lernte, sondern sich auf Pflanzen oder Regale im Hintergrund konzentrierte. Diese Hintergrundmerkmale sind oft sehr markant und können sogar dazu führen, dass der Algorithmus das Gesicht komplett ignoriert und nur den Hintergrund erkennt. Die Eingabedaten waren also zu spezifisch.

Eine einfache Möglichkeit, das Problem zu verringern, besteht darin, einen **diverseren Datensatz** (andere Landschaften, Beleuchtung, Kamerawinkel, Kopfpositionen, Zoomstufen, Haarschnitte, mehr Bilder usw.) bereitzustellen, damit dieses Phänomen seltener auftritt. Das Problem kann auch weiter reduziert werden, indem **die Daten bearbeitet** werden, um unnötige Details zu reduzieren. Alles in Graustufenbilder umzuwandeln kann die Abhängigkeit von Farben verringern, ebenso wie die Normalisierung der Sättigung aller Bilder. Geometrische Transformationen wie Skalieren, Drehen oder Spiegeln von Bildern können auch einen viel größeren Datensatz liefern, durch den er allgemeiner wird. Es ist jedoch wichtig, **deine Daten im Auge zu behalten**, wenn du verschiedenfarbige Stifte erkennen möchten, könnte es eine schlechte Idee sein, zu Graustufenbildern zu wechseln, und wenn du bestimmte Gesten erkennen möchtest, kann das Drehen eines Bildes zu Problemen führen.

Für **Underfitting** sehen wir uns das Beispiel der Vorhersage von Filmeinnahmen genauer an.

*Stell dir vor, du hast Daten von Tausenden von Filmen gesammelt (über das Genre, die Schauspieler*innen und die Einnahmen). Anschließend trainierst du dein Modell auf der Grundlage dieser Daten, stellst jedoch fest, dass die Vorhersage unabhängig von den ausgewählten Testdaten die meiste Zeit ziemlich falsch ist.*

Das bedeutet, dass dein Modell im Grunde nichts gelernt hat. Dies ist ein Beispiel für **Underfitting**, da entweder deine Trainingsdaten nicht genügend (oder relevante) Informationen enthielten, um etwas Nützliches vorherzusagen, oder deine Trainingsparameter waren weit davon entfernt.

In diesem Beispiel könnte die Einbeziehung von **mehr Merkmalen** wie dem Veröffentlichungsjahr, dem Budget und der Budgetzuweisung oder der Werbemenge dieses Problem lösen und zu einer genaueren Vorhersage führen. **Underfitting** tritt auch auf, wenn du einfach nicht über genügend Daten und daher zu wenig Informationen verfügst, die für Vorhersagen oder Klassifizierungen verwendet werden können.

Underfitting und Overfitting sind beide auffällig und normalerweise leicht zu erkennen (aber nicht unbedingt leicht zu beheben…). Als nächstes werden wir über das subtilere Problem von **Bias-Fehlern** sprechen.

Bias und Fairness

(Folien 15-19)

Während **Underfitting** und **Overfitting** technische und klar messbare Probleme sind, sind **Bias-Fehler, also Vorurteile**, viel schwerer zu erkennen. Beginnen wir mit einem realen Beispiel.

*Im Jahr 2014 versuchte Amazon, wie viele andere große Unternehmen, seinen Rekrutierungsprozess zu automatisieren, indem eine KI erstellt wurde, um Bewerbungen zu überprüfen und die talentiertesten/geeignetsten Mitarbeiter*innen zu finden. Nach einem Jahr der Entwicklung stellten sie jedoch einen großen Fehler fest: Ihr Algorithmus* **diskriminierte Frauen***! Es stellte sich heraus, dass keine böse Absicht vorlag, aber die Trainingsdaten der zuvor eingestellten Mitarbeiter*innen waren stark verzerrt in Richtung Männer, da die meisten Angestellten zu dieser Zeit männlich waren. Obwohl dieses Problem durch sorgfältige Auswahl der für das Training verwendeten Anwendungen behoben wurde, konnte nicht garantiert werden, dass der Algorithmus keine anderen, derzeit nicht sichtbaren Verzerrungen aufwies, sodass das Projekt 2018 aufgelöst wurde.*[1]

Ein weiteres Beispiel ist der Microsoft's chatbot Tay2, welcher **entzündliche und rassistische Reden von Useren von Twitter generierte***. Nach 16 Stunden wurde der Chatpot eingestellt.*[2]

Im Prinzip ist es möglich, faire Programme zu erstellen, aber je nach Thema kann es ein **sehr schwieriges Problem** sein. Das Problem reicht von Service-Bots, die Dialekte nicht erkennen, bis hin zu Klassifizierungsalgorithmen, die People of Color aufgrund begrenzter oder exklusiver Trainingsdaten nicht als Menschen erkennen. Diese Probleme sind nicht nur technischer, sondern auch ethischer Natur, da sie die Frage aufwerfen, welche Entscheidungen wir Maschinen treffen lassen sollten. Das **Ethik-Modul** geht dieser Frage genauer nach. Eine weitere gute Ressource ist **dieses Video**.

Voreingenommenheit und Fairness sind derzeit umfassend erforschte Bereiche, für die es keine einfache Lösung gibt. Alles, was du tun kannst, ist, vorsichtig zu sein, wenn du Trainingsdaten erstellst, mit denen dein Modell trainieren kann. Dies spiegelt sich auch in einem in der Informatik geläufigen Sprichwort wider: „Der Algorithmus ist nur so gut wie seine Daten“ oder einfacher ausgedrückt: „Garbage in, Garbage out“.

Trainingszeit und Transfer Learning

(Folien 20-23)

Wie in den obigen Kapiteln bereits aufgezeigt, hängt die Qualität des Modells stark von den ausgewählten Daten und daher auch von der **Größe der Daten** ab. Um ein unvoreingenommenes Bildklassifizierungsmodell zu erstellen, das Objekte in verschiedenen Situationen genau erkennt, müssen Tausende oder sogar Millionen von Bildern verwendet werden, was auch die zum Trainieren des Modells benötigte Zeit von Stunden auf Wochen erhöht.

Um diesem Problem entgegenzuwirken, kann **Transfer Learning**[3] eingesetzt werden, um von den Erfahrungen bereits gut ausgebildeter (**vortrainierter**) Modelle zu profitieren. Diese Modelle konzentrieren sich normalerweise auf allgemeinere Aufgaben, wie z. B. das Auffinden von Merkmalen in Bildern, die dann entweder als **Startpunkt** für das Training oder als **Vorverarbeitungsschritt** dazu verwendet werden können, wodurch sie die Komplexität der Eingabedaten reduzieren. Im ersten Fall kann ein Netzwerk, das bereits in der Bildklassifizierung verwendet wird, manchmal als Basis verwendet werden, um ein Netzwerk zu trainieren, um einen anderen Satz von Bildern zu klassifizieren (z. B. von der Erkennung von Waldbränden in Satellitenbildern bis zur Erkennung von Brüchen in Röntgenbildern). Im zweiten Fall kann in einem ersten Schritt ein Netzwerk verwendet werden, das Merkmale wie Kanten oder geometrische Muster erkennen kann, und das neue Netzwerk kann auf diese Merkmale anstelle der Rohbilddaten trainiert werden. In beiden Fällen kann nicht nur die Einarbeitungszeit drastisch reduziert, sondern auch die Qualität des Ergebnisses gesteigert werden, insbesondere wenn mit sehr kleinen Stichprobenumfängen wie 50 Bildern gearbeitet wird.

Die nächste Übung verwendet solche vortrainierten Netzwerke, um Bildklassifizierungsnetzwerke innerhalb des Browsers innerhalb von Sekunden bis Minuten zu trainieren, wobei nur wenige hundert Bilder für gute Ergebnisse erforderlich sind.

Material

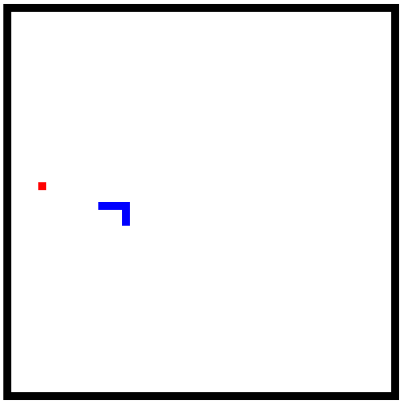
- SL - Reliance on data.pdf

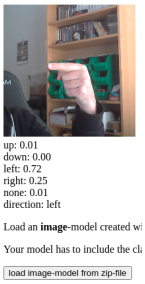
Referenzen

1. https://www.theguardian.com/technology/2018/oct/10/amazon-hiring-ai-gender-bias-recruiting-engine
2. https://www.theguardian.com/technology/2016/mar/24/tay-microsofts-ai-chatbot-gets-a-crash-course-in-racism-from-twitter
3. https://machinelearningmastery.com/transfer-learning-for-deep-learning/

Game controller

In dieser Übung trainieren die Schüler*innen ein Echtbild-Klassifizierungsmodell, um verschiedene Richtungsbefehle zu erkennen. Dieses Modell wird dann verwendet, um das bekannte Spiel "Snake" zu steuern und zu spielen.





Das Ziel dieser Übung ist es, dass die Schüler*innen ihr eigenes Echtzeit-Bildklassifizierungsmodell trainieren. Während dieses Prozesses treten viele der zuvor besprochenen Probleme auf und die Schüler*innen müssen versuchen, ihr Modell so robust wie möglich zu machen. Um die Schwierigkeiten in realen Anwendungen weiter zu demonstrieren, sollte die verwendete Kamera die Richtung/Position ändern, um Probleme in ihren Trainingssets (wie Hintergrundobjekte) hervorzuheben. Am Ende haben die Schüler*innen ein funktionierendes Modell, das Anweisungen versteht und zur Steuerung des Spiels Snake verwendet werden kann (oder für sehr fortgeschrittene Schüler*innen für jede anderen Anwendung, die sie verwenden **TensorFlow** in).

Die für diese Übung erforderliche Zeit kann stark variieren, von 30 Minuten für einen kurzen Test bis zu mehreren Stunden, wenn die Schüler*innen ein gut funktionierendes (allgemeines) Modell erstellen möchten.

Übung / Game Controller

Ziele

Die Schüler*innen sind in der Lage...

- ...ein echtes ***SL***-Bildklassifizierungsmodell zu trainieren
- ...ihr Wissen darüber anzuwenden, wie man die richtigen Daten erhält
- ...zu erkennen, dass es eine der Hauptschwierigkeiten in ***SL*** ist, dem Modell beizubringen, „das Richtige“ zu tun

Möglichkeiten und Grenzen

Jetzt, da die Schüler*innen ein gutes Verständnis davon haben, wie Supervised Learning funktioniert und was die Probleme davon sind, ist es an der Zeit, alles zusammenzufassen und sich reale Anwendungen anzusehen. Während die Grenze zwischen verschiedenen Methoden des maschinellen Lernens heutzutage ziemlich verschwommen sein kann (da zum Beispiel klassisches ***SL*** durch Reinforcement Learning verbessert werden kann), geht es in diesem Kapitel darum, den Einsatz von ***SL*** in verschiedenen Anwendungen und seine Grenzen zu erkennen.

Dieses Kapitel hat die Form eines Quiz mit (hauptsächlich) **richtigen oder falschen** Fragen auf **diesen Folien**. Die Schüler*innen können live teilnehmen, indem sie Handzeichen oder farbige/nummerierte Karten verwenden. Bei dem Quiz sollte es nicht nur darum gehen, die richtigen Antworten zu finden, sondern die Probleme **zu verstehen** und auch **zu argumentieren**, warum andere Antworten auch richtig sein könnten. Daher wird ermutigt, die Antworten nicht nur zu zeigen, sondern in einer kurzen Diskussion darüber zu sprechen. Einige Fragen enthalten zuvor nicht behandelte Themen, um den Schülern*innen die Möglichkeit zu geben, zu zeigen, wie gut sie ihr Wissen auf unbekannte Bereiche übertragen können.

Ziele

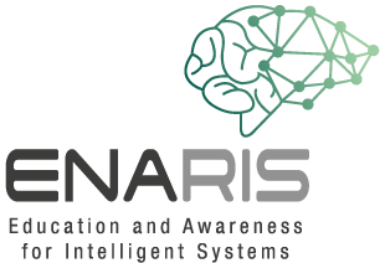
Die Schüler*innen sind in der Lage...

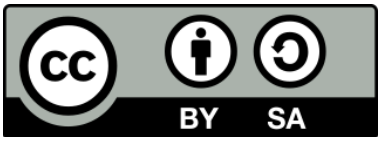
...zu erklären, was ***SL***-Lernen kann und was nicht

...zu erkennen, welche Anwendungen ***SL*** verwenden können

Material

- SL - Quiz.pdf


ENARIS
Education and Awareness
for Intelligent Systems


BY SA


EUROPEAN UNION


Interreg
Austria-Hungary 2014-2020
European Union – European Regional Development Fund


TU
Graz


mobilis
interaktív kiállítási központ


OESTERREICHISCHE
COMPUTER GESELLSCHAFT
AUSTRIAN
COMPUTER SOCIETY