

Duale Hochschule Baden-Württemberg Mannheim

Projektarbeit

Fashion-MNIST

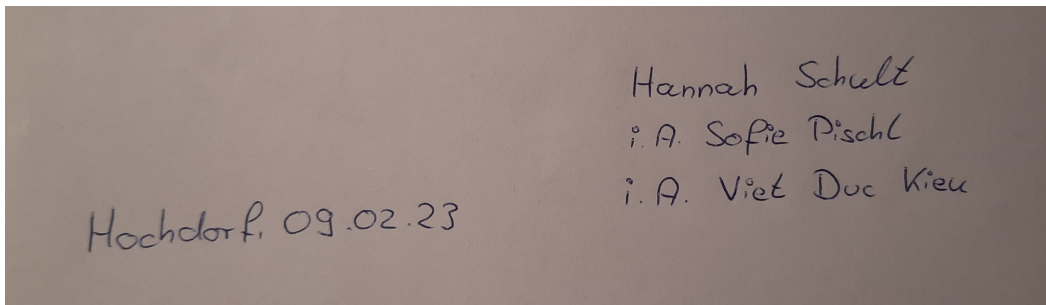
Studiengang Wirtschaftsinformatik

Studienrichtung Data Science

Verfasser(in):	Hannah Schult, Sofie Pischl, Viet Duc Kieu
Matrikelnummer:	5373022, 3943911, 9588548
Kurs:	WWI-20-DSB
Studiengangsleiter:	Prof. Dr. Bernhard Drabant
Bearbeitungszeitraum:	14.11.2022 – 10.02.2023

Ehrenwörtliche Erklärung

Wir versichern hiermit, dass wir die vorliegende Arbeit mit dem Titel "*Fashion-MNIST*" selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben. Wir versichern zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.



Hochdorf, 09.02.23

Hannah Schult
i. A. Sofie Pischl
i. A. Viet Duc Kieu

Ort, Datum

Hannah Schult, Sofie Pischl, Viet Duc Kieu

Inhaltsverzeichnis

Abbildungsverzeichnis	iii
1 Einleitung	1
1.1 Hintergrund und Motivation	1
1.2 Zielsetzung	1
1.3 Vorgehensweise	2
2 Technische Grundlagen	3
2.1 Daten	3
2.2 Verwendete Algorithmen und Modelle	5
2.2.1 DecisionTreeClassifier	5
2.2.2 RandomForestClassifier	5
2.2.3 XGBClassifier	6
2.2.4 Convolutional Neural Network	6
2.2.5 Object Detection	7
2.3 Streamlit Framework	8
3 Konzeptionierung	9
3.1 Verwendete Algorithmen und Modelle	9
3.2 Anwendung Streamlit	13
4 Evaluation	14
4.1 Ergebnisse	14
4.2 Herausforderungen	14
5 Zusammenfassung	17
5.1 Fazit	17
5.2 Ausblick	17
Literaturverzeichnis	18

Abbildungsverzeichnis

2.1	Beispiel Fashion MNIST ¹	4
3.1	Evaluierung Decision Tree nach Hyperparameter Tuning	10
3.2	Evaluierung Random Forest Classifier nach Hyperparameter Tuning	11
3.3	Evaluierung XGB Classifier nach Hyperparameter Tuning	11
3.4	Evaluierung der 3 CNN Modelle	12
3.5	Confusion Matrix CNN	12
3.6	Bildvorverarbeitung in Streamlit	13
4.1	Beispiel Ergebnis eigene Objekterkennung	15

1 Einleitung

1.1 Hintergrund und Motivation

Im Bereich Data Science ist der MNIST (Modified National Standards of Technology) Datensatz nicht wegzudenken. Er beinhaltet Graustufenbilder der Schreibschrift-Ziffern Null bis Neun, die von Hand geschrieben wurden. Der Datensatz soll dazu dienen, einen Maßstab für Machine Learning Algorithmen zu schaffen, also diese zu benchmarken. Da oft Daten in Datensätzen ungleich verteilt sind oder andere Defizite aufweisen, ist es wichtig, einen qualitativen Datensatz zu haben, auf dem Modelle getestet werden können.

Dieser MNIST Datensatz ist jedoch für die aktuellen Machine Learning Algorithmen nicht komplex genug, sodass er 2017 von Zalando mit einem Datensatz, der Bilder von Kleidungsstücken enthält, dem Fashion MNIST ersetzt wurde. Auch dieser soll eine erste Anlaufstelle für Data Scientists bieten, um klassifizierende Machine Learning Algorithmen zu prüfen¹. Dabei erfordert die höhere Komplexität der Bilder des Fashion-MNIST Datensatzes genauere Algorithmen, sodass die Qualität der Vorhersage auf dem Fashion MNIST Datensatz ein besseres Indiz für dessen Güte ist.

Des Weiteren können mit MNIST trainierte Modelle auch unter Eingabe neuer, selbst erstellter Bilder von Kleidungsstücken verwendet werden, um diese zu klassifizieren.

1.2 Zielsetzung

Im Rahmen des Seminars soll ein DecisionTree Classifier, RandomForestClassifier, ein XGB-Classifier und ein Convolutional Neural Network genutzt werden, um die Kleidungsstücke zu klassifizieren. Diese Modelle werden dann anhand ihrer Genauigkeit miteinander verglichen. Dabei wird eine Genauigkeit von mindestens 90% angestrebt, um eine präzise Klassifikation der Kleidungsstücke zu gewährleisten. So soll das Modell mit der besten Performance gefunden werden.

Dieses Modell wird dann in einer Web-Anwendung bereit gestellt, in der neue Daten in Form von Bildern von Kleidungsstücken eingegeben und klassifiziert werden können.

¹Vgl. Renee 2021.

1.3 Vorgehensweise

Dazu wird sich zunächst mit dem Datensatz vertraut gemacht. Anschließend werden zuerst die traditionellen Machine Learning Algorithmen DecisionTreeClassifier, RandomForestClassifier und XGBClassifier auf den Daten angewandt und verglichen. Im nächsten Schritt wird ein CNN (Convolutional Neural Network) Modell trainiert und mit den Algorithmen aus dem vorherigen Schritt verglichen.

Um mehrere Modelle anwenden zu können, wird eine Anwendung mit dem Framework Streamlit entwickelt. Diese Anwendung soll es ermöglichen, Fotos aufzunehmen und an die Modelle zu übergeben, um sie anschließend zu klassifizieren und das jeweilige Kleidungsstück zu identifizieren.

2 Technische Grundlagen

2.1 Daten

Der Fashion-MNIST Datensatz ist nach dem Schema des MNIST (Modified National Standards of Technology) Datensatzes erstellt. Der MNIST Datensatz beinhaltet 70.000 Bilder handschriftlich geschriebener Ziffern von Null bis Neun, die in die jeweiligen Klassen Null bis Neun unterteilt sind. Der MNIST Datensatz wurde 1998 von LeCun basierend auf dem MNIST Datensatz, der aufgrund von ungleich verteilten Daten im Trainings- und Testdatensatz zum Benchmark von Machine Learning Modellen ungeeignet ist, veröffentlicht¹. Die Klassifikation von handgeschriebenen Ziffern ist für moderne Machine Learning Algorithmen jedoch keine große Herausforderung. CNNs (Convolutional Neural Networks) können auf dem MNIST Datensatz eine Genauigkeit von 99.7% erreichen, die Ziffern können also fast perfekt vorhergesagt werden. Auch einfachere Machine Learning Algorithmen erzielen 97%. Die Qualität kann also nur schwer unterschieden werden, da die Differenzen gering sind².

Um einen neuen Datensatz zum Vergleich von Algorithmen bereit zu stellen, hat Zalando Research 2017 einen Datensatz mit Bildern von Kleidungsstücken veröffentlicht, den Fashion-MNIST Datensatz. Dieser basiert auf dem Sortiment der Zalando Webseite. Es werden Bilder verwendet, die die Artikel von vorne zeigen, wobei keine weißen Kleidungsstücke verwendet werden, da diese zu wenig Kontrast haben würden. Die Bilder werden dann auf 28x28 Pixel runter skaliert und zu einem 8 Bit großen Graustufenbild konvertiert (Abbildung 2.1). Auch dieser Datensatz wird dann in Test- und Trainingsdatensatz unterteilt. Für den Trainingsdatensatz werden zufällig 6.000 Bilder aus jeder Klasse gewählt, sodass der Trainingsdatensatz 60.000 Bilder beinhaltet. Alle Bilder sind mit einem Label versehen, das eine der Kategorien „T-shirt/Top“, „Trousers“, „Pullover“, „Dress“, „Coat“, „Sandal“, „Shirt“, „Sneaker“, „Bag“ oder „Ankle Boot“ abbildet. Die Daten werden dann nach den Labels sortiert abgespeichert, sodass die Daten komprimiert und einfacher verwendet werden können³.

⁰Vgl. *MNIST handwritten digit database*, Yann LeCun, Corinna Cortes and Chris Burges 2023

¹Vgl. Xiao, Rasul und Vollgraf 2017.

²Vgl. ZalandoResearch 2023.

³Vgl. Xiao, Rasul und Vollgraf 2017.

Beispielhafte Bilder des Fashion-MNIST und zugeordnete Label sind in der nachfolgenden Abbildung (Abbildung 2.1) zu sehen.

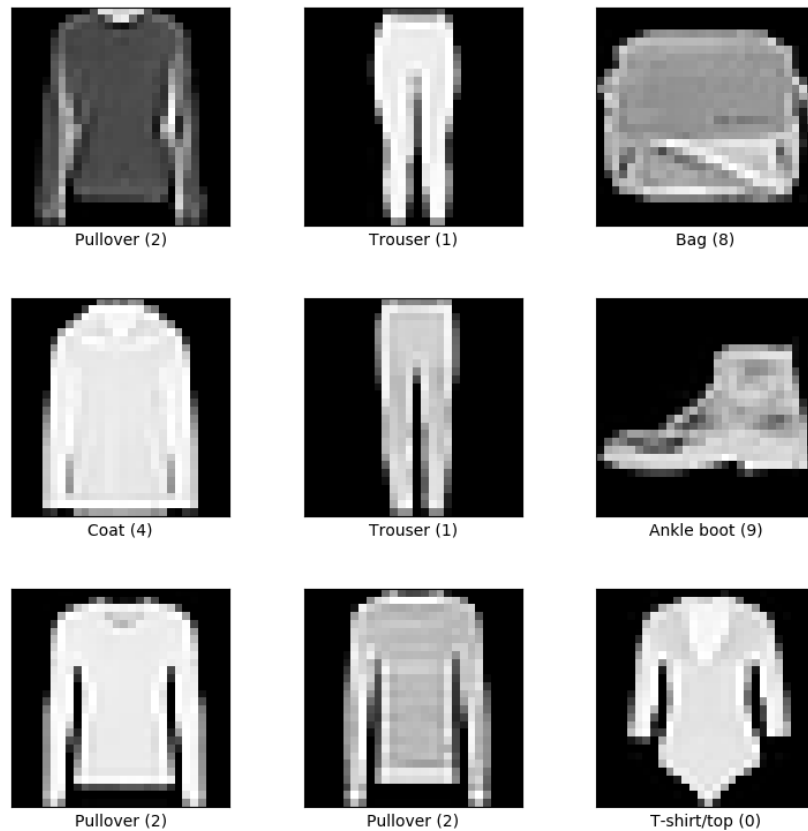


Abbildung 2.1: Beispiel Fashion MNIST⁴

⁴Vgl. *MNIST Dataset 2023*

2.2 Verwendete Algorithmen und Modelle

Während der Entwicklung wurden insgesamt vier Modelle für den Leistungsvergleich ausgewählt. Im Folgenden werden die Hintergründe der verwendeten Modelle kurz erläutert.

2.2.1 DecisionTreeClassifier

Entscheidungsbäume gehören zu den supervised learning Verfahren und sind eine nicht parametrische Methode. Den DecisionTreeClassifier kann man sowohl bei Regressions- als auch bei Klassifizierungsproblemen anwenden. Der Algorithmus nutzt die Trainingsdaten, um aus ihnen einfache Entscheidungsregeln zu erlernen. Anhand dieser wird ein Modell abgeleitet, welches den Wert der Zielvariable (in diesem Anwendungsfall die richtige Kleidungsvorhersage) vorhersagt. Je tiefer der Baum ist, desto komplexer sind die Entscheidungsregeln und desto genauer wird das Modell. Ist der Baum allerdings zu tief overfittet das Modell schnell.

Vorteile sind, dass die Entscheidungsbäume gut zu visualisieren sind und die Daten wenig Vorverarbeitung benötigen, da der Algorithmus mit kategorischen sowie numerischen Variablen arbeiten kann. Zusätzlich ist der Rechenaufwand nicht zu hoch. Nachteile sind neben dem möglichen Overfitting bei zu tiefen Bäumen, dass die Bäume auch zu Bias neigen, falls die Klassen nicht ausgeglichen sind. Da im weiteren Verlauf weitere Entscheidungsbaum basierte Algorithmen verwendet wurden (Random Forest Classifier und XGB Classifier), wurde auch der normale Decision Tree Classifier genutzt, um die Unterschiede in den Ergebnisse sehen zu können⁵

2.2.2 RandomForestClassifier

Der Random Forest Classifier gehört zu den supervised Lernverfahren und besteht aus mehreren unkorrelierten Entscheidungsbäumen. Bei einer Klassifizierung entscheidet jeder Baum einzeln, die Klasse mit den meisten Stimmen wird vorhergesagt. Ein Vorteil ist, dass der Random Forest Classifier eine höhere Genauigkeit gegenüber dem normalen Entscheidungsbaum hat. Durch eine Randomisierung vermeidet das Modell zudem das Auftreten von Overfitting. Zudem besitzt der RandomForestClassifier einen niedrigen Bias und eine

⁵Vgl. 1.10. *Decision Trees* o. D.

niedrige Varianz. Zusätzlich besitzt er viele Hyperparameter, die getunt werden können, um so die Genauigkeit der Vorhersage verbessern zu können.

Da das Modell besser als der Decision Tree Classifier ist und eine kurze Trainingsdauer besitzt, wurde dieser Algorithmus für die Problemstellung angewandt⁶.

2.2.3 XGBClassifier

XGBClassifier ist eine Implementierung von XGBoost, welche auf dem Konzept des Gradienten-Boosting-Verfahrens basiert. Es handelt sich dabei um ein Ensemble-Lernverfahren, bei dem mehrere schwache Modelle zu einem stärkeren Modell kombiniert werden, um eine höhere Vorhersagegenauigkeit zu erreichen. XGBClassifier ist speziell für Klassifikationsprobleme optimiert und nutzt eine optimierte Version des Gradienten-Boosting-Verfahrens, die eine schnellere Konvergenz und bessere Vorhersagegenauigkeit ermöglicht.

Während des Trainingsprozesses werden mehrere Entscheidungsbäume erstellt und optimiert, wobei jeder Baum eine Korrektur für die Vorhersagen des vorherigen Baums bereitstellt. Am Ende des Trainingsprozesses werden die Vorhersagen aller Bäume kombiniert, um eine endgültige Vorhersage für jede Beobachtung zu erhalten. XGBClassifier unterstützt sowohl lineare als auch nicht-lineare Modelle und es ist möglich eine große Anzahl von Hyperparametern zu optimieren, um die Vorhersagegenauigkeit zu verbessern.

Eine weitere Stärke von XGBClassifier ist seine parallelisierte Implementierung, die es ermöglicht, auf großen Datensätzen effizient zu trainieren. Aus den genannten Gründen wurde XGBClassifier für den Fashion-MNIST Datensatz ausgewählt. Das Modell kann komplexe, nicht-lineare Beziehungen zwischen den Merkmalen und der Zielklasse sehr gut erkennen.

2.2.4 Convolutional Neural Network

Convolutional Neural Networks (CNNs) sind ein wichtiger Typ von tiefen neuronalen Netzen, die für die Analyse von grid-ähnlichen Daten wie Bildern entwickelt wurden. Ein CNN besteht aus mehreren Schichten, jede mit einer spezifischen Aufgabe. Die erste Schicht, die Convolutional Layer, führt eine Convolution-Operation durch, um lokale Merkmale im Bild zu erkennen. Die Pooling Layer reduzieren die Größe des Bilds, indem sie lokale Maxima oder Mittelwerte berechnen. Die Fully Connected Layer verbinden jedes Neuron der

⁶Vgl. Fardeen 2021.

vorherigen Schicht mit jedem Neuron in der nächsten Schicht. Eine Activation Function wird verwendet, um eine Aktivierungsentscheidung zu treffen, ob das Neuron aktiviert oder deaktiviert wird. Das Verfahren der Backpropagation wird verwendet, um die Gewichte und Bias-Werte im CNN anzupassen, indem es Fehler minimiert, die von den Vorhersagen des Modells gemacht werden. Ein häufiges Problem bei neuronalen Netzen ist das Overfitting, bei dem ein Modell zu gut auf die Trainingsdaten passt und daher auf Testdaten schlechter generalisiert. CNNs haben sich als wertvolle Instrumente für Bilderkennung und -verarbeitung erwiesen und werden häufig in Anwendungen wie Bildklassifikation, Gesichtserkennung, Objekterkennung und natürliche Sprachverarbeitung eingesetzt.

2.2.5 Object Detection

Um eine bessere Performance zu erzielen, sollen die an das Modell übergebenen Bilder ähnlich denen des Fashion-MNISTs sein. Dazu sollen auch auf unruhigem Hintergrund Kleidungsstücke erkannt werden, um sie dann so anzupassen, dass sie von den Modellen möglichst gut verarbeitet werden können. Zur Erkennung der Kleidungsstücke kann Object Detection genutzt werden.

Object Detection ist ein Teilbereich von Computer Vision und Bildverarbeitung und dient dazu, spezielle Objekte in digitalen Bildern oder Videos zu identifizieren. Ein weit verbreitetes Beispiel der Object Detection ist die Gesichtserkennung. Dazu werden spezielle Features erlernt, die vom Modell dann auch in anderen Bildern erkannt werden können⁷.

Zur Erkennung von Objekten gibt es derzeit das Konvolutionale Neuronale Netzwerk YOLO (You Only Look Once), ein einmaliges Überprüfungsmodell (single-shot detection model), das zur schnellen Erkennung diverser Objekte genutzt werden kann. Objekte können in Echtzeit erkannt werden, sodass das Modell sehr effizient ist und in verschiedenen Bereichen eingesetzt werden kann⁸.

Es werden verschieden große Netzwerke zur Verfügung gestellt, die je nach Bedarf verwendet werden können. Dabei wird ein begrenzendes Rechteck (bounding box) um das jeweilige erkannte Objekt gezeichnet, dieses wird auch benannt. Dabei wird ein Bild in ein Gitter aus kleinen Zellen aufgeteilt, wobei jede Zelle eine Vorhersage für den eigenen Bereich bildet. Das Modell lernt dabei Merkmale, die zur Erkennung von Objekten wichtig

⁷Vgl. Liu et al. 2018.

⁸Vgl. Tao et al. 2017.

sind und grenzt die Objekte dann anhand dessen ab⁹.

2.3 Streamlit Framework

Streamlit ist ein Open-Source-Framework, das in Python entwickelt wurde. Es wurde im Jahr 2019 veröffentlicht und ist eine einfache und schnelle Möglichkeit, Web-Oberflächen für ML-Modelle zu entwickeln. Das Framework wurde aus vier Gründen ausgewählt.

Der erste Vorteil von Streamlit gegenüber anderen Frontend-Frameworks, wie React.js oder Vue.js, besteht darin, dass es in Python entwickelt wurde. Dies ermöglicht es, Machine-Learning-Modelle direkt in die Anwendung zu integrieren, ohne dass eine zweischichtige Anwendung mit einer Frontend-Schicht in JavaScript und einer Backend-Schicht in Python entwickelt werden muss. Dadurch lässt sich viel Zeit bei der Entwicklung der Anwendung sparen.

Ein weiterer Vorteil ist die Benutzerfreundlichkeit. Streamlit stellt responsive Webkomponenten bereit, die direkt in der Anwendung verwendet werden können. Durch diese kann die Anwendung deshalb auf jedem Gerät verwendet werden. Darüber hinaus hat Streamlit einen einfachen und intuitiven Syntax. Dies ermöglicht die Entwicklung von Benutzeroberflächen auch ohne tiefgreifende Kenntnisse über die Webentwicklung.

Der nächste Vorteil ist, dass Streamlit ein Open-Source-Framework ist. Dadurch kann Streamlit für jeden Zweck kostenlos verwendet werden. Des Weiteren lässt sich der Quellcode modifizieren, sodass dieser an seine eigene Bedürfnisse angepasst werden kann. Dies ist von Vorteil, wenn die Anwendung für einen produktiven und kommerziellen Einsatz vorgesehen ist.

Der letzte Grund ist kostenloses Hosting und Skalierbarkeit. Streamlit Community Hosting ermöglicht bis zu 1 GB kostenloses Anwendungshosting. Alle Änderungen auf Github werden direkt an die Produktivumgebung übertragen, sodass diese direkt einsehbar sind. Da die Infrastruktur von Streamlit verwaltet wird, skaliert die Anwendung automatisch und läuft auch unter hoher Last einwandfrei.

⁹Vgl. Redmon und Farhadi 2018.

3 Konzeptionierung

Der MNIST Fashion Datensatz wird zunächst eingelesen. Dann wird ein Array mit den Labels erstellt, da die Label in den Daten nur durch Zahlen von Null bis Neun beschrieben sind. Um die Modelle trainieren zu können, werden die Bilder von den Labels getrennt. Die Bilder werden auf die Variable X gespeichert und die Labels, also die Zielvariablen, auf Y.

3.1 Verwendete Algorithmen und Modelle

Die im Kapitel 1 genannten Modelle wurden immer auf die gleiche Art und Weise auf die Daten trainiert und angewandt. Zuerst wurde das Modell jeweils definiert, dann wurde das Modell mit den Trainingsdaten gefittet und anschließend wurden die Vorhersagen auf den Testdaten getroffen.

Als nächstes wurden jeweils die Genauigkeiten auf den Trainingsdaten und den Testdaten berechnet, um die Modelle so vergleichen zu können. Das Hyperparameter Tuning wurde jeweils mit Random Search durchgeführt. Mit Grid Search wären eventuell nochmal bessere Ergebnisse erzielt worden, allerdings wäre dies auch sehr viel rechenaufwändiger gewesen, weswegen das Tuning mit Random Search gemacht wurde.

accuracy	0.8104				
	precision	recall	f1-score	support	
0.0	0.76	0.79	0.77	1000	
1.0	0.97	0.93	0.95	1000	
2.0	0.70	0.74	0.72	1000	
3.0	0.80	0.86	0.83	1000	
4.0	0.69	0.72	0.71	1000	
5.0	0.91	0.87	0.89	1000	
6.0	0.62	0.51	0.56	1000	
7.0	0.82	0.89	0.85	1000	
8.0	0.93	0.92	0.93	1000	
9.0	0.89	0.88	0.88	1000	
accuracy			0.81	10000	
macro avg	0.81	0.81	0.81	10000	
weighted avg	0.81	0.81	0.81	10000	

Abbildung 3.1: Evaluierung Decision Tree nach Hyperparameter Tuning
Quelle: eigene Aufnahme

Die Ergebnisse der Decision Tree Classifier Evaluierung zeigen, dass das Modell auf manchen Klassen besonders gute Vorhersagen trifft. Die Klassen 1, 5, 8 (Hosen, Sandalen, Taschen) sagt das Modell mit jeweils mindestens 90 % Präzision vorher. Vor dem Hyperparameter Tuning hatte das Modell eine Genauigkeit von ungefähr 80 % und nach dem Tuning von etwa 81 %, also keine drastische Verbesserung. Das kommt aber wahrscheinlich daher, dass für das Tuning nur RandomSearch, aufgrund der schnelleren Laufzeit, genutzt wurde.

accuracy 0.8802					
	precision	recall	f1-score	support	
0.0	0.81	0.86	0.83	1000	
1.0	0.99	0.97	0.98	1000	
2.0	0.80	0.80	0.80	1000	
3.0	0.88	0.92	0.90	1000	
4.0	0.79	0.87	0.83	1000	
5.0	0.97	0.94	0.95	1000	
6.0	0.76	0.60	0.67	1000	
7.0	0.91	0.92	0.92	1000	
8.0	0.95	0.97	0.96	1000	
9.0	0.93	0.95	0.94	1000	
accuracy			0.88	10000	
macro avg	0.88	0.88	0.88	10000	
weighted avg	0.88	0.88	0.88	10000	

Abbildung 3.2: Evaluierung Random Forest Classifier nach Hyperparameter Tuning
Quelle: eigene Aufnahme

Die Ergebnisse der Random Forest Classifier Evaluierung zeigen, dass das Modell zusätzlich zu den Klassen 1, 5, 8 (Hosen, Sandalen, Taschen) auch die Klassen 7 und 9 (Sneaker und Stiefeletten) mit jeweils mindestens 90 % Präzision vorhersagt. Das Modell erreichte sowohl vor wie auch nach dem Hyperparametertuning ungefähr eine Genauigkeit von 88%.

accuracy 0.910375					
	precision	recall	f1-score	support	
0	0.84	0.90	0.87	815	
1	0.99	0.98	0.98	808	
2	0.85	0.84	0.85	794	
3	0.90	0.93	0.91	787	
4	0.86	0.90	0.88	781	
5	0.99	0.96	0.97	790	
6	0.80	0.71	0.75	806	
7	0.94	0.96	0.95	814	
8	0.97	0.98	0.97	794	
9	0.95	0.96	0.96	811	
accuracy			0.91	8000	
macro avg	0.91	0.91	0.91	8000	
weighted avg	0.91	0.91	0.91	8000	

Abbildung 3.3: Evaluierung XGB Classifier nach Hyperparameter Tuning
Quelle: eigene Aufnahme

Die Ergebnisse der XGB Classifier Evaluierung zeigen, dass das Modell bis jetzt das beste Modell ist. Zusätzlich zu den Klassen 1, 5, 7, 8 und 9 wird nun auch die Klasse 3 (Kleid) mit jeweils mindestens 90 % Präzision vorhergesagt. Das Modell erreichte vor dem Hyperparametertuning ungefähr eine Genauigkeit von 90 % und danach ungefähr 91 %.

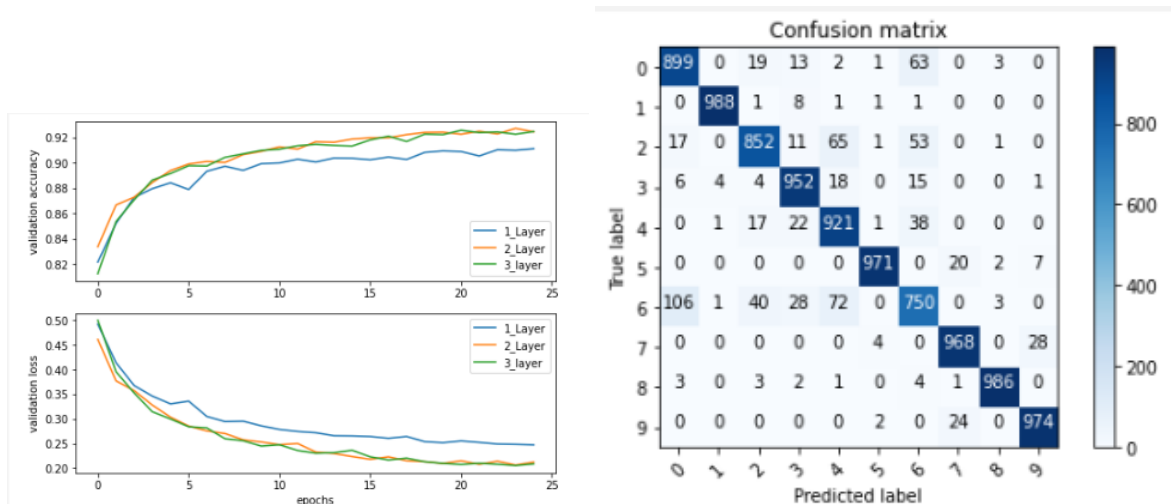


Abbildung 3.4: Evaluierung der 3 CNN Modelle
 Quelle: eigene Aufnahme

Abbildung 3.5: Confusion Matrix CNN
 Quelle: eigene Aufnahme

Insgesamt wurden drei verschiedene CNNs für das Klassifizierungsproblem trainiert. In der Abbildung 3.4 ist erkennbar, dass das erste CNN (blau) eine Genauigkeit von etwa 90 % hat und das zweite sowie dritte CNN (gelb und grün) jeweils bei einer etwa 92 %-igen Genauigkeit liegen. Es wurde sich dann für das zweite CNN Modell entschieden und in der Confusion Matrix in Abbildung 3.5 ist ersichtlich, wie das Modell auf den unterschiedlichen Klassen performt hat. Das CNN verwechselt die Klassen 0 und 6 häufiger, da das ein mal die Klasse T-shirt/ Top und einmal Shirt ist. Ansonsten funktioniert die Klassifizierung gut.

Anhand der Modelle ist erkennbar, dass das Modell XGB Classifier bei den Machine Learning Modellen das beste Modell in diesem Vergleich war. Insgesamt war aber eins der Convolutiona Neural Networks mit über 92 % Genauigkeit das beste Modell.

3.2 Anwendung Streamlit

Die Webanwendung wurde mit Streamlit entwickelt, um die ML-Modelle verwenden zu können. Die Anwendung ist in der Lage, Bilder aufzunehmen und diese anschließend an ML-Modelle zu übermitteln. Vor dem Übermitteln müssen jedoch mehrere Schritte ausgeführt werden.

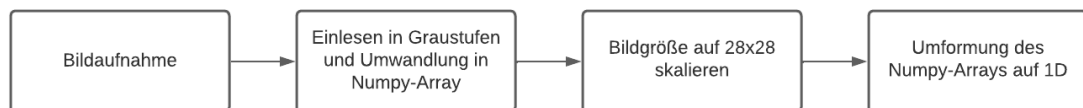


Abbildung 3.6: Bildvorverarbeitung in Streamlit

Der erste Schritt besteht darin, das Bild mit der Streamlit-Funktion „`st.camera_input`“ aufzunehmen. Diese Funktion liefert ein Objekt der `UploadedFile`-Klasse zurück, welches eine Unterklasse von `BytesIO` ist und daher ein „dateiähnliches“ Objekt darstellt¹.

Anschließend wird das Bild mithilfe von OpenCV in ein Numpy-Array konvertiert und in Graustufen umgeformt. Dies ist erforderlich, da alle Bilder im Fashion-MNIST Datensatz in Graustufen sind und die Modelle auf diese Daten trainiert wurden.

Um eine effiziente Verarbeitung der Eingabedaten durch die Modelle zu gewährleisten, wird das aufgenommene Bild auf eine Größe von 28x28 skaliert. Die Anpassung an diese Größe ist erforderlich, da alle Bilder im Fashion-MNIST-Datensatz eine Größe von 28x28 aufweisen. Nach der Skalierung hat das Daten-Format eine Dimension von (28, 28, 1), wobei 28 die Höhe und Breite des Bildes beschreibt und 1 die Anzahl der Farbkanäle angibt. Wenn es sich um ein Farbbild handeln würde, würde es stattdessen drei Farbkanäle geben, da jedes Pixel durch die Kombination von drei Farbwerten (Rot, Grün und Blau) dargestellt wird.

Schlussendlich wird das Numpy-Array in ein eindimensionales Format konvertiert, um es an die Modelle zu übergeben. Dieser Konvertierungsschritt ist jedoch nur für Modelle wie `DecisionTreeClassifier` und `RandomForestClassifier` erforderlich. Beim CNN-Modell kann die Dimension der Eingabedaten festgelegt werden, ohne sie zu konvertieren. Es ist jedoch wichtig, dass bei allen Modellen die Dimension der Eingabedaten konsistent ist.

¹Vgl. *Streamlit Documentation* 2023.

4 Evaluation

4.1 Ergebnisse

Grundsätzlich funktioniert die Klassifizierung von Kleidungsstücken, sofern diese vor einem weißen Hintergrund fotografiert werden, relativ sicher.

Wie im Kapitel Zielsetzung beschrieben, wurde die Performance der Modelle DecisionTree Classifier, RandomForestClassifier, XGBClassifier und Convolutional Neural Network auf den Fashion-MNIST Daten evaluiert und anschließend verglichen. Dabei wurde die angestrebte Genauigkeit von 90 % von den Modellen XGB Classifier und dem CNN erreicht. Das CNN war erreichte insgesamt das beste Ergebnis.

Die Web-Anwendung, in der die Modelle bereitgestellt werden, wurde entwickelt.

4.2 Herausforderungen

Falsche Vorhersage

Das erste auftretende Problem bei der Entwicklung war, dass am Anfang nur die Klasse „Bag“ vorhergesagt wurde. Zu diesem Zeitpunkt wurde der Random Forest Classifier in der Anwendung genutzt. Das Problem zu diesem Zeitpunkt war, dass die Fotos alle einen sehr unruhigen Hintergrund hatten. Da die Bilder von der Anwendung runter skaliert werden, um dem Format des MNIST Datensatzes zu entsprechen, waren es Bilder im 28x28 Format, welche zwischen Kleidungsstück und Hintergrund nur wenig Kontrast hatten. Durch die viereckige Form des Bildes wurde angenommen, das deshalb jedes mal "Bag" vorhergesagt wurde, da diese im weitesten Sinne auch eine leicht viereckige Form haben.

Um diesen Fehler zu beheben wurden zwei Lösungsansätze entwickelt. Der aufwändigere Lösungsvorschlag ist eine Object Detection auf dem Foto durchzuführen bevor das gefundene Objekt oder mehrere gefundene Objekte durch die Modelle klassifiziert werden. So wird sichergestellt, dass das Kleidungsstück vor dem Hintergrund nicht verschwindet.

Der zweite Lösungsvorschlag ist simpler. Wenn das Kleidungsstück vor einem hellen Hintergrund gemacht wird, wie zum Beispiel einer weißen Wand, entsteht genügend Kontrast, um auch ohne Object Detection das Kleidungsstück richtig zu klassifizieren.

Bildauflösung

Insgesamt war auch der Fashion-MNIST Datensatz von Zalando eine Herausforderung. Die Bilder sind alle schwarz-weiß und haben nur eine Auflösung von 28x28, was sehr gering ist. Ein Datensatz mit größeren Bildern wäre ein großer Vorteil gewesen.

Object Detection

Die Umsetzung der Object Detection wurde zunächst ohne zusätzliche Hilfe versucht umzusetzen. Diese selbst implementierte Objekterkennung erkannte jedoch mehr Objekte, als tatsächlich vorhanden waren.

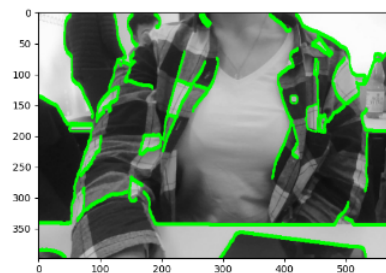


Abbildung 4.1: Beispiel Ergebnis eigene Objekterkennung

Um ein besseres Ergebnis bei der Erkennung von Objekten zu erzielen wurde dann im zweiten Versuch eine Object Detection mit YOLO5 umgesetzt.

Im vorliegenden Fall wird die Version YOLOv5n genutzt, um Kleidungsstücke zu erkennen. Diese werden dann entlang der begrenzenden Box zugeschnitten, um den Hintergrund zu minimieren.

Zur Erkennung von Kleidungsstücken wird das Modell trainiert. Dazu muss ein Datensatz gefunden werden, der eine ausreichende Anzahl an Trainingsdaten aufweist. Da hier ein Datensatz mit 160 Bildern gefunden und verwendet wurde, ist das Modell nur sehr unpräzise und erkennt Kleidungsstücke unzureichend.

Deshalb wird davon abgesehen, die Objekterkennung in die endgültige Anwendung mit aufzunehmen.

Datenformat für die Modelle

Um die Modelle in Streamlit effektiv einzusetzen, müssen diese aus dem Jupyter Notebook exportiert werden. Anfänglich wurden die Modelle mithilfe der pickle-Bibliothek exportiert, jedoch stellte sich dies als unzureichend heraus, da es beim Entpacken der Modelle zu Fehlern kam. Um dies zu vermeiden, werden die Modelle mithilfe der Funktion `st.cache()` in der Anwendung zwischengespeichert. Wenn die Modelle jedoch im falschen Format gespeichert werden, kann es ebenfalls zu Fehlern kommen. Um diese Probleme zu lösen, wurden die Modelle in ihrem nativen Datenformat gespeichert: XGBoost im txt-Format, CNN im h5-Format und die anderen Modelle im pickle-Format.

5 Zusammenfassung

5.1 Fazit

Insgesamt hat das Convolutional Neuronale Netz die höchste Genauigkeit erzielt, aber auch das XGBoost Modell erbringt eine gute Performance. Dabei ist vor allem beim RandomForestClassifier die Herausforderung, dass zunächst alle Bilder aufgrund des unruhigen Hintergrunds als „Bag “ klassifiziert werden. Die angestrebte Lösung mit Object Detection zu arbeiten, muss aufgrund dessen verworfen werden, dass kein passender Datensatz zur Verfügung steht. Es wurde sich zunächst dazu entschieden, dass die Bilder für die Anwendung vor einem weißem Hintergrund aufgenommen werden müssen, damit sie mit den Bildern aus dem Fashion-MNIST Datensatz vergleichbar sind. Außerdem stellt die niedrige Auflösung von 28x28 Pixeln, sowie das Datenformat der Modelle eine Herausforderung dar. Trotzdem kann eine funktionierende Anwendung zur Klassifikation von Kleidungsstücken bereitgestellt werden.

5.2 Ausblick

Ein erster Schritt, um die Kleidungserkennungssoftware in Zukunft zu verbessern, ist eine gut funktionierende Objekterkennung zu implementieren. Dann sind auch keine Vorgaben zur Fotoaufnahme (Bild vor weißer Wand aufnehmen) nötig. Zusätzlich können mit einer Objekterkennung auch mehrere Kleidungsstücke je Foto klassifiziert werden. Um solch eine Objekterkennung umzusetzen, muss erst ein neuer Kleidungsdatensatz mit höherer Bildauflösung erstellt werden, da die 28x28 Bilder aus dem Fashion-MNIST Datensatz zu schwach aufgelöst sind für diesen Use-Case.

Literaturverzeichnis

- 1.10. *Decision Trees* (o.D.). URL: <https://scikit-learn.org/stable/modules/tree.html>.
- Fardeen, Afham (Sep. 2021). *Random Forest Classifier in Python Sklearn with Example*. URL: <https://machinelearningknowledge.ai/python-sklearn-random-forest-classifier-tutorial-with-example/>.
- Liu, Li et al. (2018). „Deep Learning for Generic Object Detection: A Survey“. In: *CoRR* abs/1809.02165. arXiv: 1809.02165. URL: <http://arxiv.org/abs/1809.02165>.
- MNIST Dataset* (2023). URL: <https://www.engati.com/glossary/mnist-dataset> (besucht am 16.01.2023).
- MNIST handwritten digit database*, Yann LeCun, Corinna Cortes and Chris Burges (2023). URL: <http://yann.lecun.com/exdb/mnist/> (besucht am 16.01.2023).
- Redmon, Joseph und Ali Farhadi (2018). *YOLOv3: An Incremental Improvement*. DOI: 10.48550/ARXIV.1804.02767. URL: <https://arxiv.org/abs/1804.02767>.
- Renee, Tracy (26. Dez. 2021). *Fashion MNIST as an alternative to the MNIST dataset*. URL: <https://ai.plainenglish.io/fashion-mnist-as-an-alternative-to-the-mnist-dataset-38cd7e8a0ede> (besucht am 16.01.2023).
- Streamlit Documentation* (2023). URL: https://docs.streamlit.io/library/api-reference/widgets/st.camera_input (besucht am 09.02.2023).
- Tao, Jing et al. (2017). „An object detection system based on YOLO in traffic scene“. In: S. 315–319. DOI: 10.1109/ICCSNT.2017.8343709.
- Xiao, Han, Kashif Rasul und Roland Vollgraf (2017). „Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms“. In: *CoRR* abs/1708.07747. arXiv: 1708.07747. URL: <http://arxiv.org/abs/1708.07747>.
- Zalandoresearch (2023). *GitHub - zalandoresearch/fashion-mnist: A MNIST-like fashion product database. Benchmark*. URL: <https://github.com/zalandoresearch/fashion-mnist> (besucht am 16.01.2023).