Erhöhung der Zuverlässigkeit von Machine Learning Anwendungen mittels Ensemble-Methoden

Bachelorarbeit von Magnus Senfter
Tag der Einreichung: 29. April 2022

1. Gutachten: Prof. Dr.-Ing. Jürgen Adamy
2. Gutachten: Manuel Hirschle
Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

REGELUNGSMETHODEN UND ROBOTIK rmr

Fachbereich Elektrotechnik und Informationstechnik

Erklärung zur Abschlussarbeit gemäß §22 Abs. 7 und §23 Abs. 7 APB der TU Darmstadt

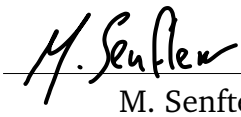
Hiermit versichere ich, Magnus Senfter, die vorliegende Bachelorarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Fall eines Plagiats (§38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß §23 Abs. 7 APB überein.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, 29. April 2022



M. Senfter

Zusammenfassung

Die bessere Verarbeitung von großen Datenmengen, steigende Rechenleistung und technische Neuentwicklungen ermöglichen einen immer vielseitigeren Einsatz von Anwendungen des Maschinellen Lernen. Es finden sich in nahezu allen Bereichen der Gesellschaft sinnvolle Einsatzmöglichkeiten. Für Anwendungen in der sicherheitskritischen Infrastruktur, wie zum Beispiel in der Medizin, muss die Zuverlässigkeit und die Robustheit der Anwendungen garantiert sein, da fehlerhafte Prädiktionen im schlimmsten Fall zum Tod von Patienten führen können. Das Ziel dieser Bachelorthesis ist eine Verbesserung der Zuverlässigkeit und der Robustheit von Klassifikatoren durch den Einsatz von Ensemble Methoden (Voting und Blending). Dazu wird das Klassifikationsergebnis anhand von Parametern, wie der Accuracy und dem Recall, für sechs unterschiedliche Klassifikationsaufgaben bewertet.

Für alle verwendeten Datensätze konnte das Klassifikationsergebnis durch Ensemble Methoden verbessert werden, wobei Blending, mittels logistischer Regression als Meta-Learner, die besten Ergebnisse erzielen konnte. Bei der Untersuchung der Robustheit der Klassifikatoren konnte anhand der implementierten Fault Injections aufgezeigt werden, dass Ensemble Methoden robuster gegenüber Störungen im Dateninput sind als ein einzelner Klassifikator. Bei starken Störungen oder Adversarial Attacks, die mehrere Basisklassifikatoren zur Fehlprädiktion verleiten, bieten Ensemble Methoden keinen adäquaten Schutz. Deshalb sollten weitere Verfahren wie Adversarial Training oder Rauschunterdrückungsverfahren angewendet werden, um die Robustheit zu maximieren.

Abstract

Due to Big Data, increased computing power and new technical developments, machine learning applications are spreading more and more into all areas of society. For applications in safety-critical infrastructure such as medicine, the reliability and robustness of the applications must be guaranteed since incorrect predictions can lead to the death of patients in the worst case. This bachelor thesis aims to improve the reliability and robustness of classifiers by using ensemble methods (voting and blending). For this purpose, the classification result is evaluated using parameters such as the accuracy and recall for six different classification tasks.

The classification result could be improved by ensemble methods for all data sets used, with blending using Meta Logistic Regression as a Meta-Learner achieving the best results. When investigating the robustness of the classifiers, the implemented fault injections showed that ensemble methods are more robust to perturbations in the data input than a single classifier. In the case of strong perturbations or Adversarial attacks that cause multiple base classifiers to mispredict, ensemble methods do not provide adequate protection. Therefore, additional techniques such as Adversarial training or data denoising should be applied to maximise robustness.

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

Adv-SS Adversarial Subspace

ART Adversarial Robustness Toolbox

AWGN additiv white gauss noise

CART Classification and Regression Trees

CT Computertomographie

FGSM Fast Gradient Sign Method

FN False Negative

FP False Postive

JSON JavaScript Object Notation

KI Künstliche Intelligenz

LiTS Liver Tumor Segmentation Benchmark

ML Maschine Learning

SGD Stochastic Gradient Descent

TN True Negative

TP True Positive

Symbolverzeichnis

A beobachtetes Bild

A$_o$ Original Bild

a$_k$ Lernrate

A$_i$ Lernalgorithmus

C$_k \in \{c_1, ..., c_i\}$ Klassenlabel

dis$_{i,j}$ Disagreement Measure

e Fehlerterm

E$_{error}$ Generalisierungsfehler

E(w,b) Regularisierter-Trainingsfehler

G(x,y) Gaussian Blur

h$_i(x)$ Basisklassifikator

h$_{ML}(x)$ Meta-Learner

J($\theta, x, y)$ lineare Approximation der Lossfunktion

L Lossfunktion

n Pertubation

N Rauschprozess

net$_j$ Nettoeingabewert

o$_i$ eingehende Information

p$_{G(z)}$ Gaussian Noise

p$_{h(x)}$ Class Probabilities

P(Y|X) bedingte Wahrscheinlichkeit

p* Entscheidungsgrenze

R Regularisierungsterm

S Datensatz bestehend aus N Elementen

$\mathbf{S}_i = \{X_i, Y_i\}$ Element des Datensatzes S bestehend aus Instanz und zugehörigem Label

$\mathbf{w}_i$ Gewichtung

X Input

$\mathbf{X}_i = \{x_1, ..., x_d\}$ d-dimensionaler Merkmalsvektor

x' Adversarial Example

x Original Input

Y Output

$\mathbf{y}_i \in [y_1, ...y_k$]Prädiktionen eines Klassifikator

$\overline{Y}$ Klassenschwerpunkt

$\mathbf{Y}^*$ kritischer Diskriminanzwert

$\mathbf{y}_{final}(x)$ Vorhersage des Ensemble

z(x) systematische Komponente

σ Schwellwert

σ Standartabweichung

σ^2 Varianz

μ Mittelwert

ϵ Größe der Pertubation

β_i Gewichtung

Inhaltsverzeichnis

1. Einleitung

Seit den Anfängen der Entwicklung von Künstlicher Intelligenz (KI) in den 1950er-Jahren gab es zahlreiche bahnbrechende Erfolge im Bereich intelligenter Systeme und Anwendungen. Bereits heute begegnet uns KI oft ganz unbewusst im Alltag [11]. Aufgrund von Big Data, wachsenden Rechenressourcen und technologischen Neuentwicklungen wird sich in naher Zukunft der Einsatz von Anwendungen basierend auf KI, insbesondere dem Machine Learning (ML), in allen Bereichen unserer Gesellschaft ausbreiten. Ebenso finden sich immer mehr sinnvolle Anwendungsmöglichkeiten in den sicherheitskritischen Infrastrukturen. Zu diesen Bereichen gehören unter anderem die Medizintechnik, die Luftfahrt und das autonome Fahren.

In der Medizin wird in nahezu allen Gebieten an sinnvollen Einsatzmöglichkeiten von neuen KI-Systemen geforscht, um den Behandlungsprozess von Patienten zu verbessern. Schon heute wird KI in der Medizin oft in Assistenzsystemen eingesetzt. Sie unterstützen das Gesundheitspersonal, treffen jedoch keine eigenen, kritischen Entscheidungen. Zum Beispiel können moderne Medizinprodukte, mit KI ausgestattet, den Arzt bei einer frühzeitigen Diagnose von Erkrankungen unterstützen. Somit steigen die Chancen für eine erfolgreiche Behandlung. Es wird an autonomen Systemen geforscht, die das Potenzial haben, komplexe Entscheidungen eigenständig zu treffen. Dazu gehören Systeme, die unter anderem anhand umfassender Patientendaten autonom Therapiemaßnahmen vorschlagen und übernehmen können. [41][47]

KI bietet enorme Chancen, eine Vielzahl an Prozessen zu optimieren. Es existieren aber auch große potenzielle Risiken. Neben ethisch-moralische Fragestellungen ist die Fehlbarkeit der intelligenten Systeme zu berücksichtigen [41]. Die Vorhersagen einer intelligenten Anwendung besitzen immer ein gewisses Fehlerrisiko. Es treten neue, unbekannte Ereignisse auf, für die das eingesetzte Modell nicht adäquat trainiert wurde. Die Inputdaten können aufgrund von natürlichen Störprozessen verzerrt wahrgenommen werden. KI kann aber auch das Ziel von mutwilligen Angriffen werden. Dabei werden Daten so manipuliert, dass das Modell eine falsche Vorhersage trifft.

Die Auswirkungen einer fehlerhaften KI hängt stark von ihrem Anwendungsgebiet ab. Eine fehlerhafte KI wie die Spracherkennung eines Smartphones ist für den Anwender lästig. Sie hat jedoch keine tiefgreifenden Folgen. In sicherheitskritischen Bereichen kann eine fehlerhafte Prädiktion hingegen katastrophale Auswirkungen für Mensch und Natur haben

[27]. In der Medizin kann zum Beispiel eine fehlerhafte Diagnose einer KI im schlimmsten Fall zum Tod eines Patienten führen. Bei der Bewertung der Zuverlässigkeit einer Anwendung der sicherheitskritischen Infrastruktur wird deshalb von einem Leben- oder Tod-Szenario ausgegangen werden. Um Schäden zu vermeiden und das Potenzial dieser Technologie auszuschöpfen, sollten daher Verfahren angewendet werden, die es ermöglichen, zuverlässige und robuste ML-Anwendungen zu erzeugen.

1.1. Zielsetzung und Problemstellung

Die Zuverlässigkeit einer ML-Anwendung ist ein entscheidendes Kriterium, um diese in einer sicherheitskritischen Infrastruktur etablieren zu können. Zur Generierung zuverlässiger Klassifikatoren gibt es verschiedene Ansätze. In dieser Arbeit soll untersucht werden, ob sich durch das Anwenden von Ensemble Methoden wie dem Blending oder Voting die Zuverlässigkeit von ML-Anwendungen verbessern lässt. Dafür wird anhand von Parametern wie der Accuracy und dem Recall für sechs verschiedene Klassifikationsaufgaben analysiert, ob mithilfe der Ensemble-Verfahren ein verbessertes Klassifikationsergebnis für die Testdaten der Datensätze erzielt werden kann.

Ein weiterer Aspekt dieser Arbeit ist die Untersuchung der Robustheit der implementierten Klassifikationsverfahren gegenüber Fault Injections. Mithilfe von Adversarial Training und Rauschunterdrückungsverfahren der Daten können zwar robuste Klassifikatoren erzeugt werden, jedoch führen diese Verfahren auch zu einem Verlust von Genauigkeit. Es gilt zu untersuchen, ob durch den Zusammenschluss mehrerer Klassifikatoren durch den Einsatz von Ensemble Methoden die Robustheit der Anwendung verbessert werden kann, ohne einen Verlust an Genauigkeit zu erzeugen.

Dafür werden Testsets generiert, die Fault Injections enthalten. Die einzelnen Instanzen werden dabei durch Störprozesse verzerrt. Durch die Pertubationen in den Daten sollen auftretende Störungen wie zum Beispiel Impulsrauschen simuliert werden und die Robustheit der Klassifikatoren gegenüber diesen untersucht werden. Neben natürlichen Störungen können ML-Anwendungen auch mutwilligen Angriffen ausgesetzt sein. Mithilfe von Adversarial Attacks werden Pertubationen in den Daten erzeugt, die zu einer fehlerhaften Prädiktion führen. Wird der Umgang einer Anwendung gegenüber Adversarial Attacks untersucht, wird auch von Adversarial Robustness gesprochen.

Zur Untersuchung der Zuverlässigkeit der Klassifikatoren werden sechs Open Source Datensätze verwendet. Die Robustheit wird anhand des MNIST Digits und MNIST Fashion Datensatz untersucht. Die Vollständigkeit und die Richtigkeit der verwendeten Daten sind als gegeben anzunehmen, weshalb keine aufwendige Datenaufbereitung notwendig ist. Um die benötigte

Rechenleistung zu minimieren, bestehen die einzelnen Objekte der Datensätze aus Graustufen-Bildern mit einer niedrigen Auflösung von $28 * 28$ Pixeln. Die Daten sind standardisiert und als NumPY Array gespeichert.

1.2. Gliederung

Im folgenden Kapitel werden die wichtigsten Grundlagen des ML erläutert, sowie das Konzept des Ensemble Learning. Dabei wird insbesondere auf die Kombination von Klassifikatoren durch Voting und Blending eingegangen. Auch Ursachen für eine fehlerhafte Vorhersage eines Klassifikators werden erläutert. Abschließend werden die Begriffe Zuverlässigkeit und Robustheit im Kontext des ML definiert.

Das dritte Kapitel beschäftigt sich mit der Durchführung der Implementierung der Klassifikationsverfahren. Die genutzte Programmierumgebung und die verwendeten Datensätze werden dazu vorgestellt. Im nächsten Schritt wird die Entwicklung der Ensemble-Verfahren eingeführt, wofür die Funktionsweise und Kombinationsmöglichkeiten der implementierten Basisklassifikatoren erklärt werden.

In Kapitel 4 werden die verwendeten Parameter vorgestellt, anhand derer die Klassifikationsergebnisse bewertet werden. Des Weiteren werden die implementierten Fault Injections präsentiert, die zur Bewertung der Robustheit der Klassifikationsverfahren verwendet werden.

In Kapitel 5 werden die Klassifikationsergebnisse der durchgeführten Versuche vorgestellt.Die Zuverlässigkeit und Robustheit der implementierten Klassifikationsverfahren wird anhand der in Kapitel 4 vorgestellten Parameter bewertet. Dabei wird zuerst analysiert, welche Meta-Features sich am besten für ein Blending eignen. Anschließend werden die einzelnen Ensemble-Verfahren miteinander verglichen.

Im letzten Kapitel dieser Thesis erfolgt das Fazit, inwiefern sich Ensemble Methoden zur Verbesserung der Zuverlässigkeit und Robustheit eignen. Des Weiteren werden Ansätze vorgestellt, wie die Zuverlässigkeit und die Robustheit weiter durch Blending Verfahren gesteigert werden könnten.

2. Grundlagen - Machine Learning (ML)

ML ist ein Teilgebiet der KI und verfolgt das Ziel, basierend auf zugrunde liegenden Daten, den sogenannten Trainingsdaten, ein mathematisches Modell zu entwickeln. Das erzeugte Modell $h(x)$ besitzt die Fähigkeit, den Input X einer Zielgröße Y zuzuweisen. Eine Instanzen X, deren Zielgröße Y ermittelt werden sollen, besteht aus einem endlichen, n-dimensionalen Merkmalsvektor $X = \{x_1, \ldots, x_d\}$. Die einzelnen Dimensionen werden auch Merkmale oder Features genannt. Anhand dieser qualitativen und quantitativen Eigenschaften eines Objektes kann das Modell eindeutig seinem Output zugeordnet werden. [17][64]

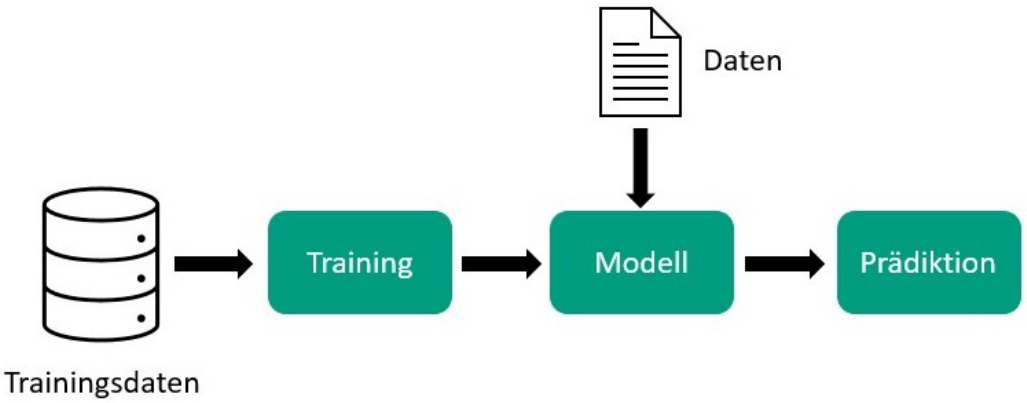


Abbildung 2.1.: Funktionsschema zur Erstellung einer ML-Anwendung (In Anlehnung an [60]).

Um eine Vorhersage über die Zielgröße Y treffen zu können, wird das Modell nicht explizit programmiert, sondern erlernt die benötigten Kenntnisse selbstständig [7]. Dazu werden spezielle Algorithmen auf die Trainingsdaten angewandt. Diese besitzen die Fähigkeit der Mustererkennung, wobei in den Daten syntaktische, statistische oder strukturelle Zusammenhänge erkannt werden. Anhand der erfassten Beziehungen wird ein möglichst realitätsnahes Modell generiert. Dadurch ist es möglich für neue, dem System unbekannte Objekte, eine Hypothese über die Zielgröße abzugeben. Dieser Prozess ermöglicht es, intelligente Anwendungen zu entwickeln, die selbstständig Probleme lösen und Entscheidungen treffen können. Wie in

Abbildung 2.1 dargestellt, soll anhand von Trainingsdaten und Training durch Algorithmen ein Modell erstellt werden, welches für unbekannte Daten eine Vorhersage treffen kann. [17]

Arten der Mustererkennung:

- **syntaktisch**: Ziel der syntaktischen Musterkennung ist es, Objekte durch die Summe ihrer Eigenschaften zu beschreiben. Dabei weisen Instanzen derselben Kategorie die gleichen Merkmale auf. Hauptproblem dieses Verfahrens ist es, eine formale Beschreibung zu entwickeln, die eine Klasse eindeutig beschreibt. [20]
- **statistisch**: Ziel der statistischen Musterkennung ist es, die Wahrscheinlichkeit einer Klassenzugehörigkeit zu berechnen. Das Objekt wird dann der Klasse mit der höchsten Wahrscheinlichkeit zugerechnet. Die einzelnen Eigenschaften eines Objektes werden durch numerische Werte abgebildet. Anhand dieses Merkmalsvektors wird eine Funktion erstellt, die jedes Objekt eindeutig einer Kategorie zuordnet. [20]
- **strukturell**: Die strukturelle Mustererkennung ist eine Kombination von Verfahren der statistischen und syntaktischen Mustererkennung. Teilmerkmale eines Objektes werden dabei oftmals anhand von statistischen Verfahren ermittelt. Anhand von syntaktischen Verfahren werden diese Erkenntnisse anschließend zusammengefasst. [20]

2.1. Supervised Learning

Je nach Anwendungsfall und der zugrunde liegenden Datengrundlage existieren verschiedene Lernverfahren, anhand derer eine ML-Anwendung entwickelt werden kann. Diese unterschiedlichen Trainingsverfahren können, wie in Abbildung 2.2 dargestellt, in drei Kategorien aufgeteilt werden [7]. In dieser Arbeit wird sich auf Anwendungsfälle beschränkt, in denen das sogenannte Supervised Learning eingesetzt wird. Diese Trainingsform wird in dieser Thesis als Einzige genauer vorgestellt.

Das Supervised Learning, auch überwachtes Lernen genannt, beschreibt eine Verfahrensart des ML, bei dem der Lernprozess aktiv vom Menschen unterstützt wird. Dabei wird zum Training des Modells ein Datensatz S mit N Elementen verwendet. Die einzelnen Elemente S_i bestehen aus einem Datentupel $S_i = \{X_i, Y_i\}$, wobei die Objekte X_i den Input für das Modell und Y_i die Zielgröße, den Output, darstellen. Diese werden als Label bezeichnet [17]. Der Output eines Modells kann sowohl diskrete als auch kontinuierliche Werte annehmen. Man spricht von Klassifikation (diskret) oder Regression (kontinuierlich).

Anhand dieser Datenpaare wird mithilfe von Algorithmen ein Modell der Form $h(x) = Y$ mit $h^* :\rightarrow Y$ erzeugt. Das trainierte Modell hat den Zusammenhang zwischen Dateninput und

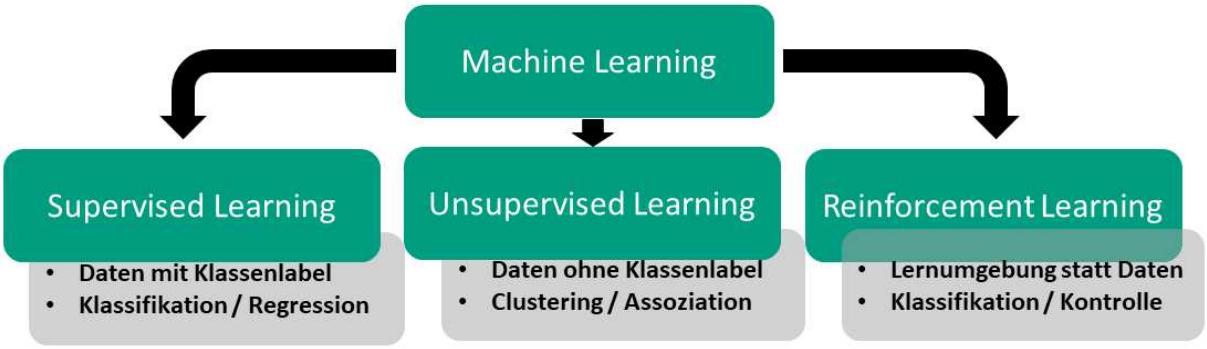


Abbildung 2.2.: Lernverfahren des Maschinellen Lernen und ihre wichtigsten Eigenschaften (In Anlehnung an [60]).

Output erlernt und kann auch für neue, unbekannte Objekte X_i eine Vorhersage über den zu erwartenden Output Y_i treffen. [31]

Essenziell für das Supervised Learning ist eine gute Aufbereitung der Daten. Die einzelnen Merkmalsvektoren müssen vollständig und fehlerfrei sein. Auch dürfen den Objekten im Trainingssatz keine falschen Klassenlabels zugewiesen werden. Diese Fehler führen dazu, dass der Zusammenhang zwischen Input und Output nicht richtig erlernt werden kann. Es wird ein fehlerbehaftetes Modell erzeugt. [31]

2.2. Klassifikation

Mit Klassifikation wird in der Datenanalyse der Prozess bezeichnet, Objekte einer kategoriellen Größe zuzuordnen. Jedes zu bestimmende Objekt besitzt bestimmte Charakteristiken. Anhand dieser qualitativen und quantitativen Merkmale kann eine Klassenzuweisung eindeutig vorgenommen werden [31]. Im ML wird dafür zur Lösung von Klassifikationsaufgaben ein Klassifikator trainiert. Also ein Modell, das den Input X einer diskreten Zielgröße Y, dem Klassenlabel C_k, zuordnet. Zur Modellgenerierung wird dabei das in 2.1 vorgestellte Trainingsverfahren des Supervised Learning angewendet. Anhand der Daten Tupels aus Merkmalsvektoren $X_i = \{x_0, \ldots, x_d\}$ und den zugehörigem Klassenlabels $C_k \in [c_1, .., c_k]$ wird ein optimales Modell trainiert, welches für einen neuen, unbekannten Dateninput X einer Klasse C_k zuordnet [42]. Klassifikationsverfahren können sowohl für binäre als auch für multinominale Probleme eingesetzt werden. [7]

- **Binäre Klassifikation**: Bei der binären Klassifikation wird zwischen zwei Klassen unterschieden. Diese Art der Klassifikation stellt einen booleschen Ausdruck dar. Ein Element

gehört entweder zu einer Klasse oder nicht. Mathematisch werden die beiden Klassen als „0 “oder „1“ codiert. Ein Beispiel dafür wäre ein Diagnosegerät, das vorhersagt, ob ein Patient an einem Tumor erkrankt ist oder nicht. Dabei würde „1 “bedeuten, der Patient wäre erkrankt und „0 “er wäre gesund. [45]

- **Multinominale Klassifikation**: Bei der multinominalen Klassifikation handelt es sich um ein Klassifikationsverfahren mit K unterschiedlichen Klassenlabels. Unsere Beispiel-KI könnte also nicht nur vorhersagen, ob der Patient erkrankt ist, sondern auch, ob der vorhandene Tumor maligne oder benigne ist. [45]

Viele der für die Klassifikation entwickelten Verfahren, wie zum Beispiel die logistische Regression oder die Support Vector Machine, sind nur zum Lösen binärer Klassifikationsaufgaben mit $C_k \in [0, 1]$ geeignet. Durch das Anwenden heuristischer Methoden kann eine Klassifikationsaufgabe mit mehr als zwei Klassen auch in eine Vielzahl an binären Probleme aufgeteilt werden. [2]

One vs Rest Verfahren

Eine Klassifikationsaufgabe mit K Klassen wird dabei in K binäre Klassifikationsaufgaben aufgeteilt, für die jeweils ein eigener Klassifikator $h_i(x)$ trainiert wird. Jeder Klassifikator $h_i(x)$ sagt dabei vorher, ob eine Instanz X zur Klasse c_i oder zu den restlichen $K - 1$ Klassen gehört. Wird zum Beispiel eine Klassifikationsaufgabe mit K Klassen mittels logistischer Regression gelöst, werden K Entscheidungsfunktionen erzeugt. Jede der Entscheidungsfunktionen gibt Auskunft über die Wahrscheinlichkeit der Klassenzugehörigkeit zu einer spezifischen Klasse c_i. Die Instanz wird der Klasse zugeordnet, deren Entscheidungsfunktion maximal ist. [13]

One vs One Verfahren

Beim One vs One Verfahren wird die Klassifikationsaufgabe mit K Klassen in $\frac{K(k-1)}{2}$ binäre Aufgaben aufgeteilt. Die $\frac{k(k-1)}{2}$ Klassifikatoren führen einen paarweisen Vergleich zwischen den Klassen durch. Per Mehrheitsentscheid wird eine finale Klasse bestimmt. Die Klasse, die am häufigsten von den Klassifikatoren vorhergesagt wurde, wird als finale Prädiktion gewählt. [13]

Bilderkennung

Ein großes Anwendungsgebiet des ML ist die Bilderkennung. Alle in dieser Thesis verwendeten Datensätze und die daraus resultierenden Modelle wurden zur Bilderkennung entwickelt. Die Anwendungen erkennen Objekte in Bildern und ordnen diese ihrer zugehörigen Kategorie zu. Das Modell erkennt Strukturen in der Farbwertverteilung der Pixel eines Bildes. Anhand dieser Muster kann sie anschließend vorhersagen, zu welcher Klasse das im Bild abgebildete Objekt gehört. [40]

2.3. Output der Basisklassifikatoren

Allgemein besteht für die meisten Anwendungen der Output eines Klassifikators $h(x)$ für eine Klassifikationsaufgabe mit k Klassenlabels aus einem k dimensionalen Vektor $h_i^1(x), ..., h_i^k(x)$. Wobei die Vorhersage $h_i^l(x)$ Auskunft darüber gibt, ob eine Instanz zur Klasse c_l gehört oder nicht [64]. Für die Werteverteilung der Vorhersagen werden häufig Crisp-Labels oder Class Probabilities eingesetzt. Für das Meta-Learning werden neben den Vorhersagen oft noch weitere Features aus den Prädiktionen abgeleitet. Deshalb werden als Output der Klassifikatoren Class Probabilities verwendet. Falls ein Klassifikator aufgrund seiner Modellstruktur nicht in der Lage ist, eine Wahrscheinlichkeitsverteilung vorherzusagen (zum Beispiel Decision Tree) werden Crisps Labels als Vorhersage genutzt. [31][64]

- **Crisp-Labels**: $h_i(x) \in \{0, 1\}$, wobei $h_i(x)$ den Wert 1 annimmt, wenn das jeweilige Label vom Klassifikator vorhergesagt wird. Entspricht $h_i(x)$ nicht dem vorhergesagten Klassenlabel, ist der Wert 0. [31][64]
- **Class Probabilities**: $h_i(x) = [0, 1]$, wobei der Output des Klassifikators $p_h(x)$ eine Wahrscheinlichkeitsverteilung der Klassenzugehörigkeiten ist. Dabei ist die Vorhersage für ein Label c_1 die A-posteriori Wahrscheinlichkeit $p_C(c_1|x)$. [31][64]

$$p_h(x) = (p_C(c_1|x), p_C(c_2|x), ...p_C(c_k|x)) \tag{2.1}$$

2.4. Ensemble Methoden

Ensemble Methoden, auch Ensemble Learning genannt, umfassen im ML Verfahren, bei denen zur Entscheidungsfindung ein Set an Klassifikatoren $h_i \in \{h_1(x), ..., h_i(x)\}$ verwendet wird. Die einzelnen Klassifikatoren h_i, auch Basisklassifikatoren oder Level 0 Classifier genannt, geben individuelle Vorhersagen y_i für eine Instanz ab. Diese Vorhersagen werden anschließend zu einer finalen Hypothese y_{final} kombiniert. Der Erfolg von Ensemble Methoden beruht auf der "Weisheit der Vielen". Es wird davon ausgegangen, dass jedes Modell bestimmte Limitationen besitzt und deshalb fehlerhafte Prädiktionen eintreten werden. Durch den Zusammenschluss einer Vielzahl von Klassifikatoren ergänzen sich die Starken und die Schwachen zu einem verbesserten Klassifikator. [64][59]

Die Implementierung eines Ensemble aus Klassifikatoren besteht im Wesentlichen aus zwei Schritten. Im ersten Schritt wird ein diverses Set an Klassifikatoren erzeugt. Diversität kann zum Beispiel durch Manipulation der Trainingsdaten geschaffen werden. Die Klassifikatoren werden dafür an unterschiedlichen Teilmengen des Datensatzes oder einer anderen Auswahl an Features trainiert. Dadurch variiert die Varianz der Modelle und es kann auch für ein

homogenes Set an Klassifikatoren, bei dem jeweils derselbe Lernalgorithmus A verwendet wird, Diversität erzeugt werden. Bekannte Verfahren, die diesen Ansatz anwenden, sind das Bagging und das Boosting. In dieser Arbeit wird der Ansatz der Verwendung eines heterogenen Sets an Basisklassifikatoren verfolgt. Dabei wird die Diversität der Basisklassifikatoren $h_i(x)$ durch das Verwenden unterschiedlicher Lernalgorithmen A_i erzeugt. Dieses Konzept wird unter anderem beim Stacking oder Blending angewandt. Ist nun ein diverses Set an Klassifikatoren erzeugt worden, muss im nächsten Schritt ein Verfahren angewendet werden, um die Vorhersagen y_i der Basisklassifikatoren zu kombinieren. [64][31]

Zur Kombination können sowohl statische als auch dynamische Verfahren genutzt werden. Bei statischen Ensemble Methoden erfolgt die Kombination nach einer festgelegten Vorschrift. Zu den am häufigsten verwendeten statischen Verfahren zählt das Voting. Dabei werden die Vorhersagen mittels Mehrheitsentscheid kombiniert. Für dynamische Verfahren wird zur Entwicklung einer Kombinationstechnik ein Meta-Learner $h_{ML}(x)$ trainiert. Dabei handelt es sich um einen weiteren Klassifikator, der mithilfe eines Lernalgorithmus aus den Vorhersagen der Basisklassifikatoren für einen Validationsdatensatz ein Kombinationsverfahren entwickelt. [53][18]

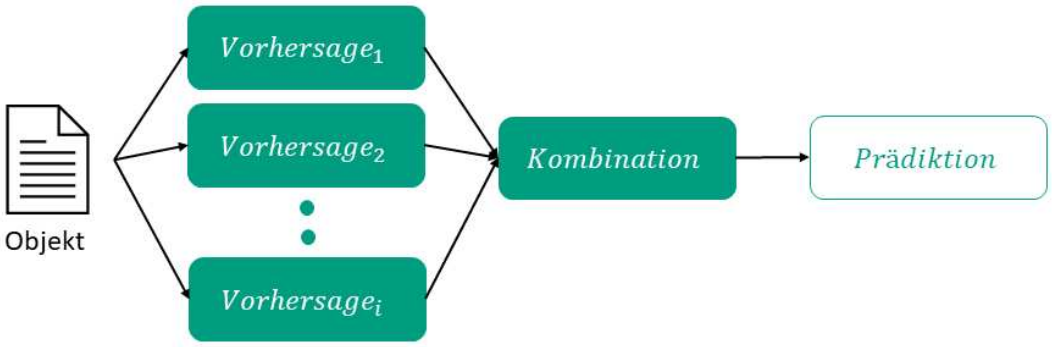


Abbildung 2.3.: Funktionsschema von Ensemble Methoden (In Anlehnung an [64]). Die Vorhersagen der Basisklassifikatoren werden zu einer Vorhersage kombiniert.

Wieso verbessert sich nun genau die Klassifikationsleistung einer Anwendung durch den Einsatz von Ensemble Methoden? Nach dem von Dietterich veröffentlichten Paper beruht der Vorteil der Kombination von Klassifikatoren auf dem statistischen, rechnerischen und repräsentativen Problem eines einzelnen Lernalgorithmus [15]. Diese drei Faktoren sind Hauptursachen, die zu fehlerhaften Hypothesen eines einzelnen Klassifikators führen und können durch den Einsatz von Ensemble Methoden abgeschwächt oder gar behoben werden können. [64]

- **Statistisches Problem**: Dem Lernalgorithmus steht oft nur eine begrenzte Menge an Trai-

ningsdaten zur Verfügung. Ist die Anzahl an Trainingsdaten im Vergleich zum möglichen Hypothesenraum zu klein, kann es beim Training vorkommen, das mehre Hypothesen die gleiche Wahrscheinlichkeit besitzen. Durch das Verwenden eines Ensembles reduziert sich aufgrund der Kombination mehrere Vorhersagen das Risiko der Wahl einer falschen Hypothese. Klassifikatoren, bei denen das statistische Problem auftritt, besitzen häufig eine hohe Varianz. [15]

- **Rechnerisches Problem**: Einzelne Klassifikatoren können in Abhängigkeit des Startpunktes bei der Suche nach einer optimalen Zielfunktion in lokalen Optima stecken bleiben. Durch ein Ensemble, deren Klassifikatoren unterschiedliche Startpunkte zur Suche nach dem Optimum verwenden, kann eine bessere Annäherung des Modells an die Realität erzeugt werden. Tritt das rechnerische Problem auf, wird von einer hohen computational Varianz gesprochen. [15]
- **Repräsentatives Problem**: Das von einem einzelnen Klassifikator erzeugte Modell kann oftmals die Realität nicht richtig abbilden. Durch den gewichteten Einsatz verschiedener Modelle ist es möglich, den Hypothesenraum zu erweitern und eine bessere Annäherung an die Realität zu schaffen. Das repräsentative Problem sorgt für einen hohen Bias eines Klassifikators. [15]

2.4.1. Voting

Voting stellt ein sehr intuitives Ensemble-Verfahren für die nominale Klassifikation dar. Dabei wird zur Entscheidungsfindung kein Training des Meta-Layer benötigt. Stattdessen löst ein Set aus T Klassifikatoren $h_i(x) = \{h_1, ..., h_t\}$ eigenständig die Klassifikationsaufgabe. Die Vorhersagen $y_i = \{y_1, ..., y_t\}$ der Klassifikatoren werden anschließend durch ein Mehrheitsvotum zu einer einzigen Vorhersage y_{final} kombiniert [31][64]. Zur Konsensfindung können verschiedene Verfahren angewandt werden:

Majority Voting

Beim Majority Voting löst jeder Klassifikator die Klassifikationsaufgabe eigenständig. Die Klassifikatoren stimmen anschließend für das von ihnen vorhergesagte Klassenlabel c_i. Als finale Vorhersage y_{final} wird das Label bestimmt, das mehr als 50% der Stimmen erhält. Erreicht keines der Klassenlabels die benötigte Anzahl an Stimmen, verwirft der Klassifikator das Ergebnis. Eine Klassifikationsaufgabe wird erfolgreich gelöst, wenn mehr als $T/2 + 1$ Klassifikatoren das korrekte Label vorhersagen. [31][64]

$$H(x) = \begin{cases} c_i & if \sum_{i=1}^{T} h_i^j(x) > \frac{1}{2} \sum_{k=1}^{l} \sum_{i=1}^{T} h_i^k(x) \\ rejection & otherwise \end{cases} \tag{2.2}$$

Uniform Voting

Uniform Voting folgt dem Prinzip der Einstimmigkeit. Die Anwendung trifft nur eine gültige Vorhersage, wenn alle Basisklassifikatoren $h_i(x)$ die gleiche Prädiktion y_i machen. Alle Klassifikatoren müssen für das gleiche Klassenlabel c_j stimmen, ansonsten wird die Vorhersage verworfen. [31]

$$H(x) = \begin{cases} c_i & if \sum_{i=1}^{T} h_i^j(x) = \sum_{k=1}^{l} \sum_{i=1}^{T} h_i^k(x) \\ rejection & otherwise \end{cases} \tag{2.3}$$

Plurality Voting

Beim Plurality Voting wird das Klassenlabel c_j als finale Vorhersage gewählt, das am häufigsten von den Basisklassifikatoren $h_i(x)$ vorhergesagt wurde. In seltenen Fällen kann es hier zu einer Stimmgleichheit zwischen den Klassenlabels kommen. In diesem Fall sollte das Ergebnis verworfen oder eine Verfahrensanweisung definiert werden. [31][64]

$$H(x) = c_{argmax \sum_{i=i}^{T} h_i^j(x)} \tag{2.4}$$

Soft Voting

Für Klassifikatoren, die Crisp-Labels als Output verwenden, können die oben genannten Verfahren zur Kombination verwendet werden. Diese Kombinationsverfahren werden auch Hard Voting genannt, da jeder Klassifikator nur eine Stimme besitzt. Anders ist es beim Soft Voting. Hier teilen die Basisklassifikatoren ihre Stimmen auf alle Klassenlabels auf. Output der Basisklassifikatoren ist dabei eine Wahrscheinlichkeitsverteilung. Zur Ermittlung der finalen Vorhersage werden die Wahrscheinlichkeiten für die einzelnen Labels summiert. Das Label mit der höchsten Gesamtwahrscheinlichkeit wird anschließend als finale Vorhersage gewählt. [31][64]

$$H^j(x) = \frac{1}{T} \sum_{I=1}^{T} h_i^j(x) \tag{2.5}$$

Weigthed Voting

In der Realität variiert die Zuverlässigkeit der verschiedenen Basisklassifikatoren oft stark. Eine gleichwertige Gewichtung der Stimmen ist aus diesem Grund nicht optimal. Stattdessen sollten die Vorhersagen von „besseren“Klassifikatoren stärkere Auswirkung auf den Entscheidungsprozess haben als die von „schlechteren“. Dieser Ansatz wird Weigthed Voting bezeichnet

und kann sowohl für Hard- als auch Soft-Voting-Verfahren eingesetzt werden. Die Vorhersagen der Klassifikatoren h_i werden mittels Gewichtungen w_i gewichtet. Dabei nimmt w_i oft einen Wert zwischen 0 und 1 an. Kunst des Weigthed Votings ist das Bestimmen einer geeigneten Gewichtung, um ein besseres Klassifikationsergebnis zu erzielen [31][64]. Zur Gewichtung der Vorhersagen werden häufig folgende Konzepte angewandt:

- **Weight per Classifier**: Bei diesem Verfahren bekommt jeder Klassifikator h_i eine Gewichtung w_i in Abhängigkeit seiner Güte. Die Gewichtung kann anhand von ermittelten Parametern wie der Accuracy oder dem Recall erfolgen. Die Gewichtungen können aber auch eigenständig festgelegt werden. Wie in Formel 2.6 beschrieben, wird das Klassenlabel als finales Label gewählt, dessen Summe aus gewichteten Vorhersagen den Mehrheitsentscheid gewinnt. [31]

$$H^j(x) = \sum_{I=1}^{T} w_i h_i^j(x) \tag{2.6}$$

- **Weight per Classifier per Class**: Oft unterscheiden sich Klassifikatoren auch in der Fähigkeit, bestimmte Label richtig vorherzusagen. Um dieses Verhalten in das Voting miteinbeziehen zu können, werden die Vorhersagen eines Klassifikators in Abhängigkeit vom vorhergesagten Klassenlabel gewichtet. Dafür kann eine labelabhängige Gewichtung w_i^j eingeführt werden. Gerade bei Klassifikationsaufgaben mit einer großen Anzahl an Klassenlabels entsteht ein enormer Implementierungsaufwand, weshalb sich dieses Verfahren nur bedingt eignet. Ein Voting mit Klassengewichtung kann wie folgt formuliert werden: [31]

$$H^j(x) = \sum_{I=1}^{T} w_i^j h_i^j(x) \tag{2.7}$$

2.4.2. Blending

Blending beschreibt ein Ensemble-Verfahren, bei dem verschiedene Modelle zur Klassifikation miteinander kombiniert werden. Im ersten Schritt wird ein diverses Set an Klassifikatoren $h(x) = \{h_1(x), ..., h_T(x)\}$ erzeugt. Die Diversität der T Klassifikatoren entsteht dabei in der Regel durch das Verwenden von unterschiedlichen Lernalgorithmen $A\ in\{A_1, ..., A_n\}$ (z.B logistische Regression, Decision Tree etc.). Die unterschiedlichen Klassifikatoren werden in diesem Kontext Basisklassifikatoren oder „Level 0 Classifier “ genannt. Die Basisklassifikatoren werden anhand desselben Datensatzes S mit $S_i = (X_i, Y_i)$ trainiert. [48][64]

Als Trainingsverfahren wird das Hold Out Verfahren verwendet. Der Datensatz wird in Trainings-, Validations- und Testdaten aufgeteilt. Wird zum Training der Klassifikatoren

eine Cross Validation verwendet, spricht man von Stacked Generalisation (Stacking). Im nächsten Schritt werden die Basisklassifikatoren mithilfe des Trainings eines weiteren Klassifikators $H_{ML}(x)$, dem Metaklassifikator kombiniert. Dieser Meta-Learner wird anhand der Vorhersagen der Basisklassifikatoren den sogenannten Metadaten trainiert. Häufig werden die Features der Metadaten noch mit aus den Vorhersagen abgeleiteten Parametern und weiteren Features ergänzt. [53][52]

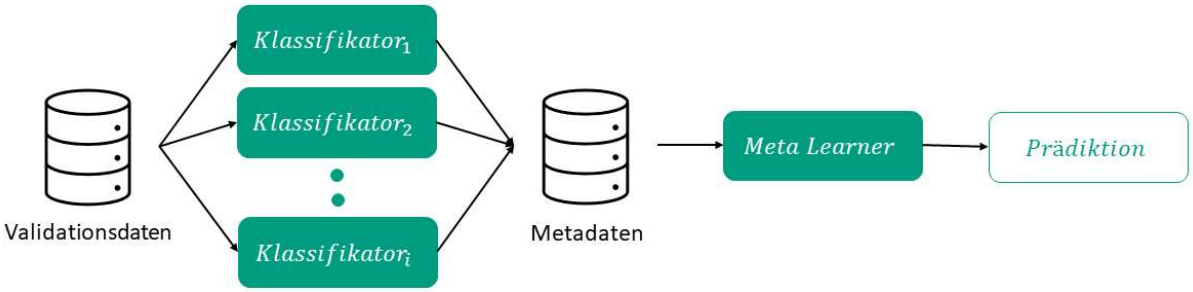


Abbildung 2.4.: Funktionsschema eines Blending-Verfahren (In Anlehnung an [53]). Anhand der Vorhersagen der Basisklassifikatoren für den Validationsdatensatz wird ein Verfahren zur Kombination der Vorhersagen trainiert.

Meta-Learning, also das Lernen aus Erfahrungen, kann im ML für folgende Zwecke eingesetzt werden:

- Mithilfe des Meta-Learners wird ein Verfahren mit dem Ziel entwickelt, den besten Klassifikator $h_i(x)$ zur Klassifizierung einer Instanz X_i zu wählen. Anhand von Meta-Features werden Bereiche bestimmt, für die unterschiedliche Klassifikatoren verwendet werden. Es wird in diesem Fall auch von „Classifier Selection“ gesprochen. [18]
- Eine weitere Möglichkeit ist die dynamische Anpassung des Bias eines Klassifikators mittels Meta-Learning. Der so erzeugte Klassifikator kann einen auf das zu klassifizierende Objekt angepassten Hypothesenraum auswählen. [18]
- In dieser Arbeit werden lediglich Blending Verfahren mit dem Ziel der Kombination der Vorhersagen untersucht. Im Kontrast zum Voting, das einer festen Entscheidungsvorschrift unterliegt, wird beim Blending eine dynamische Entscheidungsvorschrift entwickelt. Wie in Abbildung 2.4 dargestellt, wird anhand der Metadaten $\{h_i(x), c_i\}$, bestehend aus den Vorhersagen der Basisklassifikatoren für den Validationsdatensatz $\{h_1(x_i), ...h_n(x_i)\}$ und dem korrekten Klassenlabelc_i, ein Modell zur Kombination der Vorhersagen der Basisklassifikatoren entwickelt. [18]

2.5. Fehlerarten

Wie in Formel 2.8 zu sehen, setzt sich das tatsächliche Klassenlabel aus der Prädiktion $h(x)$ des Modells und einem Fehlerterm e zusammen. [34]

$$c_i = h(x) + e \tag{2.8}$$

Die aus den Trainingsdaten induktiv erlernten Modelle können also fehlerbehaftet sein. Der Fehler eines Modells, der bei der Vorhersage von Instanzen entsteht, die dem Trainingsprozess vorenthalten waren, wird auch Generalisierungsfehler E_{error} genannt. Die Berechnung des Generalisierungsfehlers E_{error} kann Formel 2.9 entnommen werden. Der Generalisierungsfehler setzt sich aus einem unreduzierbaren Fehler $Noise$ und einem Fehler, der durch die Optimierung des Modells verbessert werden kann, zusammen. Der unreduzierbare Fehler $Noise$ entsteht durch unbekannte Variablen, die im Modell nicht berücksichtigt werden können, unabhängig von der Güte des Modells. Der reduzierbare Fehler hingegen setzt sich aus $Bias^2$ und Varianz eines Modells zusammen. [34]

$$E_{error} = Bias^2 + Var + Noise \tag{2.9}$$

Bias

Der Bias ist der systematische Fehler eines Modells, also die Differenz zwischen vorhergesagten Klassenlabel y_i und tatsächlichen Wert y. Der Bias entsteht durch die erlernten Annahmen zur Klassifizierung. Kann der Klassifikator die notwendigen Zusammenhänge zur Entscheidungsfindung nicht erlernen, besitzt der Klassifikator einen hohen Bias und kann weder Trainingsdaten noch Testdaten richtig klassifizieren. [34][51][22]

$$Bias = E(y_i - y) \tag{2.10}$$

Varianz

Im ML steht häufig nur eine beschränkte Menge an Daten zum Training eines Klassifikators zur Verfügung. Deshalb können nicht alle Zusammenhänge zwischen den Merkmalen eines Objektes und seiner Klassenzugehörigkeit erfasst werden. Anhand der Varianz kann gemessen werden, wie sich die induzierte Entscheidungsregel durch Variation der Trainingsdaten ändert [34]. Kommen ähnliche Datenstrukturen zu häufig vor, kann das Modell zwar grundlegende Zusammenhänge erlernen, aber auch Rauschprozesse erfassen. Ist dieses Rauschen nun in einem unbekannten Objekt nicht enthalten, kann die Instanz nicht korrekt klassifiziert werden. Die Folge einer hohen Varianz ist, dass das erzeugte Modell zu gut auf die Trainingsdaten

angepasst ist. Die Instanzen der Trainingsdaten können fehlerfrei klassifiziert werden. Kleinste Änderungen in den Daten führen aber zur Fehlklassifikation, weshalb die Testdaten nicht korrekt klassifiziert werden können. Weist ein Klassifikator eine hohe Varianz auf, wird von Overfitting des Klassifikators gesprochen. [51][22]

$$Variance = E((y_i - E(y_i))^2 \tag{2.11}$$

Bias Variance Tradeoff

Das Klassifikationsergebnis eines Klassifikators wäre optimal, würde dieser sowohl einen niedrigen Bias als auch eine niedrige Varianz besitzen. Es hat sich jedoch gezeigt, dass ein Zusammenhang zwischen Bias und Varianz existiert. So weisen Modelle mit einem geringen Bias häufig eine große Varianz auf und Modelle mit einer großen Varianz einen geringen Bias. Bei der Implementierung eines ML Modells muss also ein Kompromiss zwischen Bias und Varianz gefunden werden. Es wird auch von Bias Variance Tradeoff gesprochen. [34][10][64]

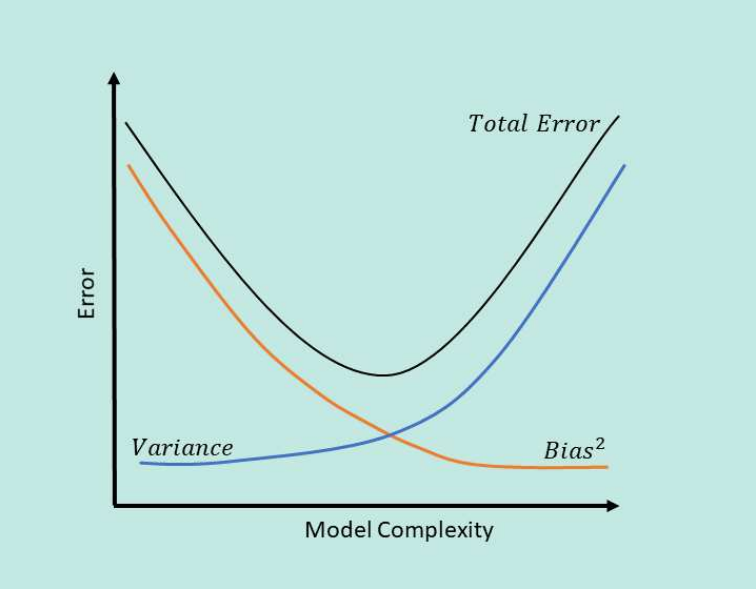


Abbildung 2.5.: Bias^2, Varianz und Total Error als Funktion der Modellkomplexität (In Anlehnung an [37]).

Wie in Abbildung 2.5 dargestellt, kann der Generalisierungsfehler minimiert werden, wenn ein Gleichgewicht zwischen Bias und Varianz gefunden wird. Erfüllt ein Klassifikator diese Eigenschaften, hat das Modell die wesentlichen Zusammenhänge in den Daten erfasst und das Rauschen in den Daten hat nur einen geringen Einfluss auf die Klassifikationsleistung. [10][64]

Abbildung 2.6.: Over-/Underfitting im Machine Learning (In Anlehnung an [6]). (a) das Modell konnte keine zufriedenstellenden Muster in den Trainignsdaten erlernen, (b) das Modell ist optimal angepasst, (c) das Modell weißt eine Überanpassung an die Trainingsdaten auf.

Overfitting

Im ML besteht beim Supervised Learning die Gefahr der Überanpassung (eng. Overfitting) des Modells. Wird das Modell zu lange trainiert oder stehen zu wenige Trainingsdaten zur Verfügung, kann das entwickelte Modell zwar die Trainingsdaten richtig klassifizieren, eine allgemeine Klassifikation von Daten ist jedoch nicht möglich. Daten, die nicht in den Trainingsdaten enthalten waren, können nicht richtig klassifiziert werden. Grund dafür ist der Fakt, dass für die Trainingsdaten oft mehr als ein mathematisches Modell existiert, welches den Zusammenhang zwischen Input und Output richtig darstellt. [16]

Es besteht nun die Gefahr, dass ein Modell gewählt wird, das zu sehr auf Ausreißer angepasst ist und deshalb unbekannte Daten nicht mehr richtig klassifizieren kann. Ein weiteres Risiko ist es, dass sich das Modell an das Rauschen in den Trainingsdaten anpasst, anstatt allgemeine Vorhersageregeln zu finden. Modelle, die eine Überanpassung aufweisen, haben in der Regel eine hohe Varianz und einen geringen Bias. [16]

Underfitting

Andererseits besteht bei zu geringem Trainingsaufwand oder unzureichenden Trainingsdaten die Gefahr der Unteranpassung (eng. Underfitting). Das Modell ist somit nicht in der Lage, einen geeigneten Zusammenhang zwischen Input und Output zu erlernen. Deswegen kann das Modell weder für die Trainingsdaten noch für die Testdaten eine hohe Genauigkeit erzielen. Weist das Klassifikationsergebnis einen hohen Bias und eine geringe Varianz auf, kann dies eine Indikation für eine Unteranpassung darstellen. [16]

Ziel ist es, ein Modell zu entwickeln, das weder über- noch unterangepasst ist. Das Modell soll die Zusammenhänge zwischen Input und Output der Trainingsdaten richtig darstellen können und für unbekannte Daten eine richtige Vorhersage treffen.

2.6. Zuverlässigkeit

Die Zuverlässigkeit ist definiert als die Wahrscheinlichkeit, dass ein Gerät seine vorhergesehene Funktion zufriedenstellend für einen spezifischen Zeitraum unter spezifischen Betriebsbedingungen erfüllt [43]. Gerade in Bereichen, bei denen durch fehlerhafte Prädiktionen existenzielle Gefahren für Mensch und Natur entstehen können, ist die Zuverlässigkeit von intelligenten Systemen eine der wichtigsten Eigenschaften, die über die Anwendbarkeit in der Praxis entscheiden.

Die Frage, wie zuverlässig eine ML-Anwendung sein muss, lässt sich nicht allgemein beantworten. Die benötigte Zuverlässigkeit einer Anwendung ist stark abhängig von den möglichen Folgen einer fehlerhaften Prädiktion. Allgemein ist der Einsatz von ML-Anwendungen nur erstrebenswert, wenn die zu erledigende Aufgabe besser gelöst wird als von menschlichen Experten des jeweiligen Fachgebietes. Nach dem Paper von S.Saria und A Subbaswamy existieren drei grundlegende Prinzipien, um die Zuverlässigkeit im ML sicherzustellen[43].

- Die **Fehler Prävention** verfolgt das Ziel, das Auftreten von Fehlern beziehungsweise die Häufigkeit von Fehlern zu reduzieren. Es sollen vorbeugend häufige Fehlerquellen identifiziert werden und auf dieser Grundlage Methoden angewandt werden, die diese korrigieren [43].
- **Fehleridentifizierung und Zuverlässigkeitsüberwachung** sollen fehlerhafte Vorhersagen einer ML-Anwendung frühzeitig erkannen und so deren schadhafte Auswirkungen reduzieren [43].
- Eine regelmäßige **Wartung** gewährleistet die andauernde Zuverlässigkeit einer ML-Anwendung. So können unteranderm neu aufgetretene Sicherheitslücken geschlossen werden [43].

Die in dieser Thesis angewandten Methoden haben dabei den Zweck, die Fehler Prävention einer ML-Anwendung zu verbessern. Die Zuverlässigkeit der verschiedenen Klassifikationsverfahren dabei anhand der in Kapitel 4.1 vorgestellten Verfahren bewertet.

2.7. Robustheit

Robustheit beschreibt in der Informatik die Fähigkeit einer Software, die Funktionsfähigkeit der Anwendung trotz auftretender Störungen zu erhalten. Sie ist dadurch ein integraler Bestandteil eines zuverlässigen Systems. Häufige Fehlerquellen sind dabei unter anderem Hardware- oder Softwaredefekte, aber auch fehlerhafte Dateneingaben. Gerade im ML kann ein Klassifikator nicht für alle möglichen Dateneingaben trainiert werden. In dieser Thesis beschränken wir uns deswegen auf diesen Aspekt der Robustheit. Es wird untersucht, wie die Klassifikationsverfahren auf Pertubationen in den Inputdaten reagieren. Die Daten können durch natürliche Rauschprozesse gestört sein, aber auch mutwillige Manipulationen durch den Menschen sind nie auszuschließen. [21]

Die Robustheit einer ML-Anwendung wird dabei häufig durch Detektion oder Prävention von Fehlern gewährleistet, wobei sich in dieser Arbeit auf die Prävention von Fehlern fokussiert wird. Ein gängiger Ansatz ist dabei das sogenannte Adversarial Training. Zusätzlich zu den erhobenen Trainingsdaten wird das Modell anhand von vorsätzlich manipulierten Daten trainiert. Dieses Verfahren kann die Anwendung auf häufig auftretende Störungen vorbereiten und es ist möglich, robuste Klassifikatoren zu erzeugen. Der Einsatz von Adversarial Training führt zu einem Verlust der Genauigkeit des Klassifikators [49]. In dieser Arbeit wird deshalb untersucht, ob Robustheit auch ohne den Verlust an Genauigkeit des Klassifikators erreicht werden kann. Dies soll durch den Einsatz eines diversen Ensembles an Klassifikatoren und der geschickten Kombination der Vorhersagen erreicht werden.

3. Implementierung der Klassifikatoren

In diesem Kapitel werden die verwendete Programmierumgebung und die verwendeten Datensätze vorgestellt. Des Weiteren werden die Funktionsweise der Basisklassifikatoren und der implementierten Ensemble Methoden erläutert. Zum Lösen der Klassifikationsaufgabe wurde für alle Klassifikatoren wie folgt vorgegangen:

- Laden des Datensatzes
- Preprocessing der Daten
- Training der Basisklassifikatoren
- Trainig der Ensemble Methoden
- Implementierung der Fault Injections
- Evaluation der Klassifikatoren anhand der Testdaten und Fault Injections

3.1. Programmierumgebung

Als Programmierumgebung wird in dieser Arbeit **Jupyter Notebook** verwendet. Jupyter Notebook ist eine webbasierte, interaktive Computingumgebung. Dieses Open Source Projekt wurde für wissenschaftliche Datenauswertungen entwickelt und verwendet als Programmiersprache Python 3. Bei den Notebooks handelt es sich um JSON- Dokumente, die aus Eingabe- und Ausgabezellen bestehen.

ML benötigt oft eine sehr große Menge an Rechenleistung in Form von GPU und CPU. Da diese benötigte Hardware lokal oft nur begrenzt zur Verfügung steht, werden die verschiedenen Jupyter Notebooks mithilfe Cloud gehosteter Rechenleistung des Clouddienstes **Google Colaboraty** ausgeführt. Zur Implementierung der verschiedenen Notebooks werden zahlreiche Bibliotheken verwendet. Einige für das ML Relevante werden nun in Kürze vorgestellt:

Numpy ist eine Open Source Programmbibliothek für das numerische Computing mit Python. Diese Bibliothek ermöglicht das Verarbeiten von großen, mehrdimensionalen Arrays.

Scikit Learn auch bekannt als Sklearn ist eine Open Source Libary für die Programmiersprache Python. Es handelt sich um eine einfach anzuwendende und effiziente Toolbox für Datenanalysen. Diese Bibliothek enthält eine Vielzahl an State of the Art Algorithmen sowohl für das Supervised als auch das Unsupervised Learning. [19]

Tensorflow ist eine zum Themengebiet KI und ML entwickelte Programmbibliothek. Sie wurde von Google als Open Source Lizenz zur Verfügung gestellt. Das Framework ist plattformunabhängig und ermöglicht es auf einfache Weise neuronale Netze zu erstellen.

Matplotlib ist eine Programmbibliothek für die Programmiersprache Python. Sie ermöglicht es dem Programmierer, einfach und unkompliziert verschiedenste Daten zu visualisieren.

Adversarial Robustness Toolbox (ART) ist eine Python Toolbox zum Thema ML Security. ART unterstützt die gängigsten ML Frameworks wie Keras, Tensorflow und Scikit-Learn. Die Toolbox enthält eine Vielzahl an Evasion, Poison und Extraction Verfahren, aber auch verschiedene Verfahren zur Verteidigung gegenüber diesen Angriffen. Dies ermöglicht die Evaluation und Verteidigung von ML Modellen gegenüber Adversarial Attacks. [39]

3.2. Datensätze

Grundlage für die durchgeführten Versuche sind verschiedene Open Source Datensätze. Es werden nur Datensätze verwendet, die für die Bildklassifikation geeignet sind. Da die Daten schon aufbereitet zur Verfügung gestellt werden, ist eine Vollständigkeit und Korrektheit der einzelnen Elemente anzunehmen. Falls vorab nicht vorgenommen, werden die Werte der Features noch auf einen Bereich zwischen 0 und 1 standardisiert. Die Größe der verwendeten Datensätze variiert stark und es werden sowohl binäre als auch multinominale Klassifikationsaufgaben untersucht. Die genaue Größe des Datensatzes und die Anzahl an Labels sind in Tabelle 3.1 aufgezeigt.

Für die Implementierung und die Evaluierung der Ensembleverfahren müssen die Datensätze in drei Teile aufgeteilt werden.

- **Trainingsdaten**: Der Trainingssatz wird verwendet, um die Modelle der Basisklassifikatoren zu trainieren.
- **Validationsdaten**: Anhand der Vorhersagen der Basisklassifikatoren für die Validationsdaten werden die Meta-Learner trainiert
- **Testdaten**: Die Zuverlässigkeit der klassifikatoren wird anhand eines Testdatensatzes evaluiert. Um Overfitting auszuschließen ist es entscheidend, dass die Objekte der Testdaten nicht im Trainingssatz oder Validationssatz enthalten sind.

Tabelle 3.1.: Aufteillung der Datensätze in Trainings-, Validations- und Testdaten sowie die Anzahl an Klassenlabels.

Datensatz	Trainingsdaten	Validationsdaten	Testdaten	Gesamt	Labels
MNIST Digits	45000	15000	10000	70000	10
MNIST Fashion	45000	15000	10000	70000	10
OrganA MNIST	34581	6491	17778	58850	11
OrganC MNIST	13000	2392	8268	23660	11
Breast MNIST	546	78	156	780	2
Pneumonial MNIST	4708	524	624	5856	2

MNIST Digits

Die MNIST Datenbank (Modified National Institute of Standards and Technology database) ist eine öffentlich verfügbare Datenbank. Seit der Veröffentlichung im Jahr 1998 gehört sie zu einer der am häufigsten verwendeten Testumgebungen für das ML. Aufgrund seiner Einfachheit wird der MNIST Datensatz oft als Benchmark zur Validation von Algorithmen verwendet. [4]

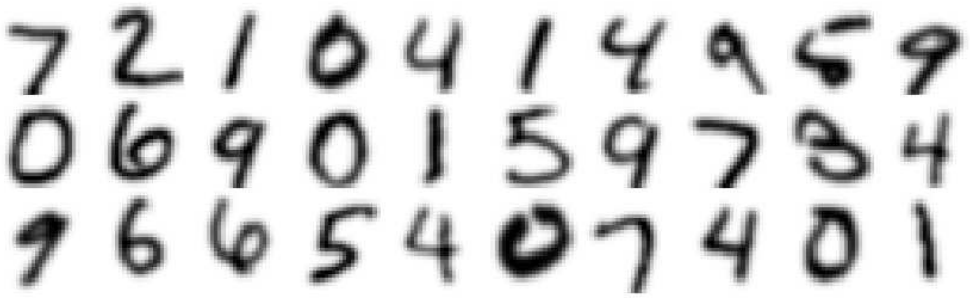


Abbildung 3.1.: Darstellung von 30 Instanzen des MNIST Fashion Datensatzes.

Der Datensatz enthält insgesamt 70000 Objekte. Davon umfassen die Trainingsdaten 60000 Instanzen. Die restlichen 10000 werden zum Testen verwendet. Jedes Objekt stellt eine handgeschriebene Ziffer mit einem Wert zwischen null und neun dar. Die Häufigkeit der verschiedenen Klassenlabels ist mit je 7000 Stück gleichverteilt. Die geschriebenen Ziffern wurden auf die gleiche Größe normalisiert, zentriert, auf eine Größe von $28 * 28$ Pixel transformiert und als Graustufenbild gespeichert. [4]

MNIST Fashion

Der Fashion MNIST Datensatz enthält Frontaufnahmen von 70000 einzigartigen Kleidungsstücken. Die Produkte stammen aus dem Sortiment von Zalando. Die Kleidungsstücke sind in zehn verschiedene Klassen eingeteilt mit 7000 Bildern pro Kategorie. Die einzelnen Objekte des Datensatzes sind als $28 * 28$ Pixel Graustufenbild gespeichert. Fashion MNIST stellt im Vergleich zum MNIST Digits Datensatz eine etwas anspruchsvollere Benchmark für die Klassifikation dar. [61]

Abbildung 3.2.: Darstellung von 30 Instanzen des MNIST Fashion Datensatzes.

MedMnist v2

Des Weiteren werden zur Evaluation der Zuverlässigkeit vier verschiedene Datensätze der MedMnist v2 Datenbank verwendet. MedMnist v2 ist eine dem MNIST Datensatz nachempfundene Datenbank biomedizinischer Bilder. Diese Datenbank enthält zwölf 2D und sechs 3D Datensätze. Die Größe der Datensätze variiert zwischen 100 und 100.000 Objekten. Die Daten sind für verschiedene Verfahren (wie binary/multi-class, ordinal regression, and multi-label) geeignet. Die enthaltenen Daten sind alle standardisiert, weshalb keine Vorverarbeitung der Daten notwendig ist. Für diese Thesis wurden vier 2D Datensätze verwendet. Die Bilder dieser Datensätze besitzen eine Größe von $28 * 28$ Pixel in Graustufen und sind als NumPY Format gespeichert. [63] Die Datensätze sind jeweils in einen Trainings-, einen Validations- und einen Testdatensatz aufgeteilt. Die Aufteilung ist in Tabelle 3.2.1 dargestellt. Es werden folgende Datensätze des MedMNIST v2 verwendet:

MedMNIST OrganA, C

Der OrganMNIST Datensatz enthält 3D Computer Tomographie (CT) Aufnahmen der Liver Tumor Segmentation Benchmark(LiTS). Zur Vereinfachung werden die Aufnahmen dieses Datensatzes in ihrer Schnittbildebene aufgeteilt. Somit stehen drei Datensätze mit axialen, koronalen oder sagittalen CT Bildern zur Verfügung. Wobei in dieser Arbeit nur die Datensätze

mit axialer und koronaler Sicht verwendet werden. Ziel der Klassifikationsaufgabe ist es, die abgebildeten Organe richtig zu klassifizieren. Zur einfacheren Verarbeitung der Bilder wurde die Hounsfield-Skala mit einem abdominalen Fenster in Graustufen umgewandelt. Die Graustufenbilder wurden auf eine Größe von $28 * 28$ Pixel reduziert und bilden eines von elf verschiedenen Organen des menschlichen Körpers ab. [63]

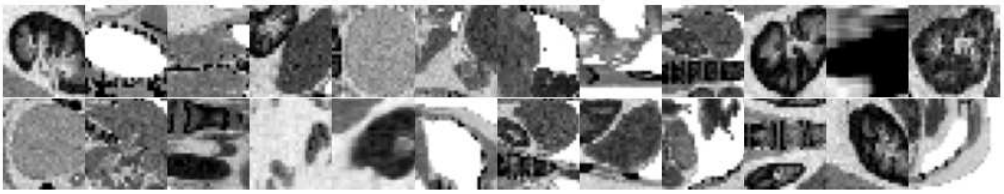


Abbildung 3.3.: Darstellung von 24 Instanzen des MedMNISTv2 OrganC Datensatzes.

MedMNIST Pneumonial

Der PneumonialMNIST Datensatz enthält 5856 pädiatrische Bruströntgenbilder. Ziel der Klassifikationsaufgabe ist es Pneumonie zu erkennen. Pneumonie ist eine Entzündung des Lungengewebes. Somit handelt es sich um eine binäre Klassifikationsaufgabe. Die Instanzen des Datensatzes werden in krank und gesund aufgeteilt. Die Originalbilder sind in Graustufen mit einer Größe von $(384 - 2916) * (127 - 2713)$ Pixeln gespeichert. Der Bildausschnitt wird anschließend zentriert und auf eine Größe von $28 * 28$ Pixeln verkleinert. [63]

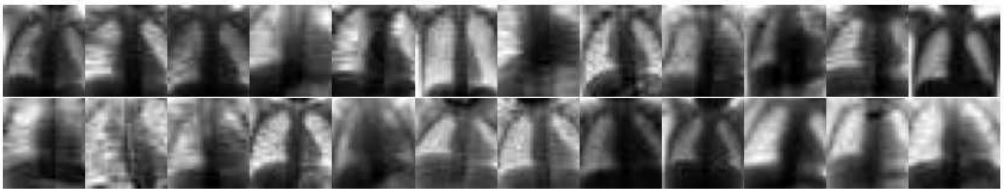


Abbildung 3.4.: Darstellung von 24 Instanzen des MedMNISTv2 Pneumonial Datensatzes.

MedMNIST Breast

Der BreastMNIST Datensatz enthält 780 Ultraschallbilder der Brust. Ziel dieser Klassifikationsaufgabe ist die richtige Detektion von Brusttumoren. Dabei werden die Bilder in drei Klassen eingeteilt: kein Tumor, benigne Tumor und maligne Tumor. Um die Klassifikationsaufgabe zu vereinfachen, werden Ultraschallbilder, auf denen kein Tumor zu erkennen ist und Bilder, auf

denen ein benigner Tumor zu erkennen ist, in eine Klasse zusammengefasst. Normale und benigne Ultraschallbilder werden als 0 und Maligne als 1 klassifiziert. Der Datensatz wird im Verhältnis von 7:1:2 aufgeteilt. Die Originalbilder haben eine Größe von 500x500 Pixel und werden auf die Größe von $28 * 28$ Pixeln verkleinert. [63]

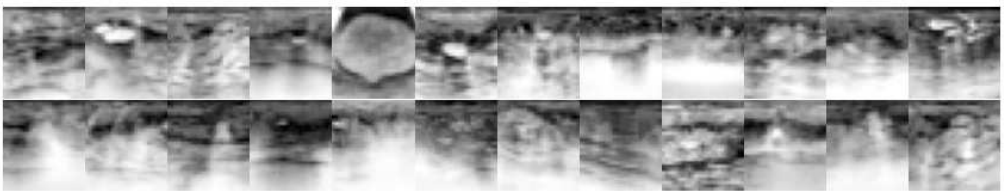


Abbildung 3.5.: Darstellung von 24 Instanzen des MedMNISTv2 Breast Datensatzes.

3.3. Klassifikationsverfahren

Das Basis Layer der Ensemble Methoden besteht aus sieben Klassifikatoren $h_i(x)$ mit $i \in \{1, 2, ..., 7\}$. Dabei handelt es sich um ein heterogenes Set an Klassifikatoren. Es werden sieben verschiedene Lernalgorithmen zur Erstellung der Modelle angewandt. Als Basisklassifikatoren werden folgende Klassifikatoren implementiert: logistische Regression, lineare Diskriminanzanalyse, Stochastic Gradient Descent, Decision Tree, Random Forest, Support Vector Machine und ein künstliches neuronales Netz. Zum Training der Klassifikatoren wurde das Hold Out Verfahren eingesetzt. Das Neuronale Netz wurde mithilfe von Tensorflow implementiert und für die Implementierung der restlichen Klassifikatoren wurde die scikit-learn Bibliothek verwendet. Die Hyperparameter der verschiedenen Verfahren sind für diese Arbeit nicht von tieferer Bedeutung und werden aus diesem Grund nicht weiter thematisiert. Die Funktionsweise der einzelnen Klassifikationsverfahren wird im folgenden Abschnitt in Kürze erläutert.

Zum Training der Klassifikatoren h_i werden folgende Annahmen getroffen:

- Es wird ein Datensatz S mit N Instanzen verwendet.
- Jede Instanz S_i setzt sich aus einem d dimensionalen Merkmalsvektor $x_i \in \{x_1, ..., x_d\}$ und dem dazugehörigen Klassenlabel y_i mit $y_i \in [y_1, ...y_k]$ zusammen.
- Ziel der Klassifikation ist es, ein optimales Modell $h_i(x)$ zu finden, das für eine unbekannte Instanz X_i das korrekte Klassenlabel y_i vorhersagt mit $y_i = h(x)$.

3.3.1. Logistische Regression

Die logistische Regression ist eine Erweiterung der linearen Regression, welche die binäre Klassifikation ermöglicht. Der Output der logistischen Regression nimmt einen kategorialen Wert an, der die jeweilige Klasse repräsentiert. Ziel des Modells ist es, anhand der logistischen Funktion die bedingte Wahrscheinlichkeit $P(Y_{=1}|x)$ für die Zugehörigkeit eines Objektes X_i zur Klasse, die mit dem Wert 1 dargestellt wird, zu berechnen. Dafür werden zuerst die Dimensionen des Merkmalsvektors x mittels Linearkombination zu einer systematischen Komponente $z(x)$ (3.1) kombiniert, wobei β_i die Gewichtung der einzelnen Merkmale darstellt. In einem iterativen Verfahren werden die Gewichtungen β_i nach der Maximum-Likelihood-Methode geschätzt, sodass eine größtmögliche Anzahl von Trainingsdaten korrekt vorhergesagt werden kann. [2]

$$z(x) = \beta_o + \beta_1 * x_1 + ...\beta_j * x_j \tag{3.1}$$

Die Wahrscheinlichkeit der Zugehörigkeit der Instanz zur Klasse 1 wird anschließend mittels der logistischen Funktion (3.2) berechnet. Die logistische Funktion ist in Abbildung 3.6 dargestellt. Sie ist eine „S“ förmige Funktion mit dem Wertebereich $[0, 1]$. Mit ihrer Hilfe kann eine Variable mit dem Wertebereich $[-\infty, \infty]$ in den Wertebereich $[0, 1]$ transformiert werden. Wobei je größer der Wert von $z(x)$, desto größer ist auch die $P(Y_{=1}|X)$ also die Wahrscheinlichkeit der Zugehörigkeit der Instanz X_i zur Klasse „1“. [2]

$$P(Y_{=1}|x) = \frac{e^z}{1 + e^z} = \frac{1}{1 + e^{-z(x)}} \tag{3.2}$$

Die Zuweisung des Klassenlabels geschieht anhand einer fest definierten Entscheidungsgrenze $p*$. Wie in Formel (3.3) dargestellt, wird eine Instanz X_i der Klasse „1“ zugewiesen, wenn die mithilfe der Entscheidungsfunktion ermittelte A-posteriori Wahrscheinlichkeit $P(Y_{=1}|x)$ größer ist als die Entscheidungsgrenze p^*. Ist der ermittelte Wert kleiner, wird das Klassenlabel „0“ vorhergesagt. Für die meisten Klassifikationsaufgaben wird eine Entscheidungsgrenze von $p* = 0.5$ gewählt. [11]

$$H(x) = \begin{cases} 1 & wenn\ p_k > p* \\ 0 & wenn\ p_k \leq p* \end{cases} \tag{3.3}$$

Mit der logistischer Regression können nur binäre Klassifikationsaufgaben gelöst werden. Mit dem One vs Rest Verfahren wird die Klassifikationsaufgabe mit K Klassen in K binäre Probleme aufgeteilt. Als finale Vorhersage wird das Klassenlabel bestimmt, dessen Entscheidungsfunktion $P_k(Y_{=1}|x)$ maximal ist. In diesem Fall wird von einer multinominalen logistischen Regression gesprochen. [2]

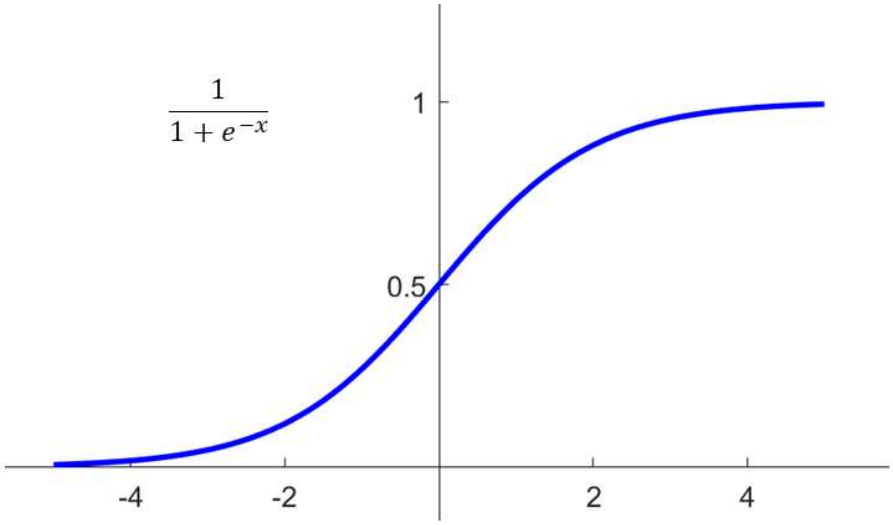


Abbildung 3.6.: Verlauf der logistischen Funktion für den Wertebereich [-5,5] (In Anlehnung an [2]).

3.3.2. Lineare Diskriminanzanalyse

Die Diskriminanzanalyse gehört zur Klasse der Struktur prüfenden Verfahren. Wie bei der logistischen Regression wird die Klassifikationsaufgabe anhand eines statistischen Verfahrens gelöst. Bei der Diskriminanzanalyse wird anhand der Diskriminanzfunktion zwischen den verschiedenen Klassen unterschieden. Bei dieser Funktion handelt es sich um eine kanonische Funktion, also einer Linearkombination aus Variablen [2]. Eine Diskriminanzfunktion hat folgenden Aufbau:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_J x_J \tag{3.4}$$

Die Diskriminanzfunktion liefert für jede Instanz X_i mit den Merkmalsvariablen $x \in \{x_1, ..., x_j\}$ eine Diskriminanzvariable Y_i. Das konstante Glied β_o und die Diskriminanzkoeffizienten der Merkmalsvariablen β_j werden anhand der zugrunde liegenden Daten so geschätzt, dass eine optimale Unterscheidung zwischen den Klassen möglich ist. [2]

Anhand der Trainingsdaten wird für jede der K Klassen ein Schwerpunkt $\overline{Y_k}$ ermittelt. Zur Ermittlung der Klassenzugehörigkeit eines Objektes X_i werden die zugehörigen Distanzen zu den Klassenschwerpunkten berechnet. Ist die Distanz einer Diskriminanzvariable zum Schwerpunkt einer Klasse niedriger als ein zuvor definierter kritischer Diskriminanzwert Y^*, gehört das Objekt X_i dieser Klasse an. [2]

Für binäre Klassifikationsaufgaben kann die Entscheidungsfindung wie folgt definiert werden:

$$H(x) = \begin{cases} 1 & wenn\ Y_i < Y^* \\ 0 & wenn\ Y_i > Y^* \end{cases} \tag{3.5}$$

In dieser Arbeit wird die lineare Diskriminanzanalyse nach R.A Fisher verwendet. Dabei wird für jede Klasse eine Diskriminanzfunktion bestimmt. Das Objekt X wird anschließend der Klasse zugeordnet, deren Diskriminanzvariable Y_k maximal ist.

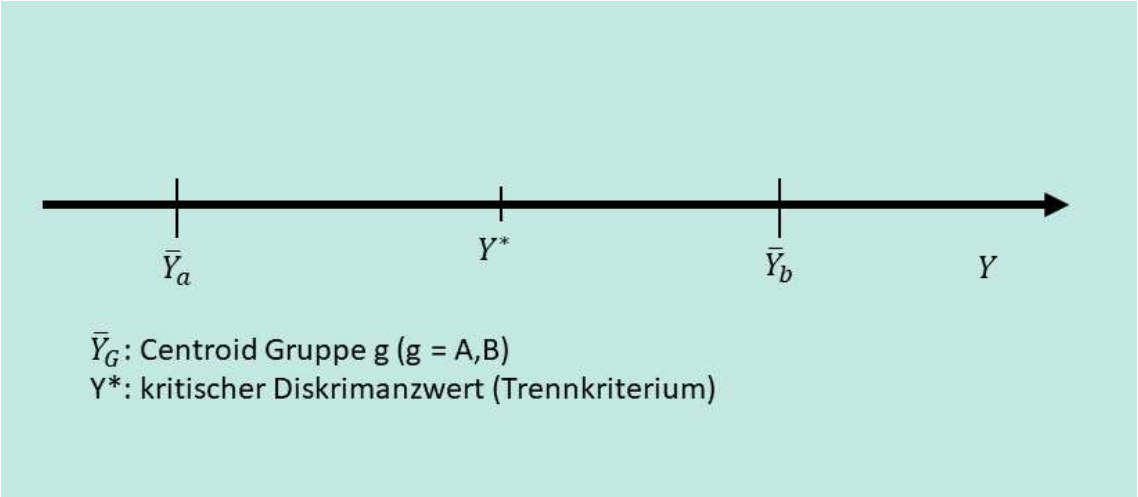


Abbildung 3.7.: Diskriminanzachse von zwei Klassen (In Anlehnung an [2]).

3.3.3. Stochastic Gradient Descent

Das Stochastic Gradient Descent (SGD) ist genau genommen kein Lernverfahren, sondern ein Solver für ein Optimierungsproblem. Es wird verwendet, um eine Zielfunktion zu optimieren. Mithilfe des SGD können verschiedene Entscheidungsfunktionen einer ML-Anwendung optimiert werden. In dieser Arbeit wird die Entscheidungsfunktion der logistischen Regression optimiert. Anstatt die Gewichtungen β_i anhand der Maximum-Likelihood-Methode zu schätzen, werden sie mithilfe des SGD bestimmt. SGD ist ein effizientes, iteratives Lösungsverfahren und wird im ML dazu eingesetzt, den Regularisierten-Trainingsfehler zu minimieren. Der Regularisierte-Trainingsfehler kann anhand von Formel 3.6 berechnet werden. [62][38]

$$E(w,b) = \frac{1}{n} \sum L(y_i, h(x_i) + \alpha * R(w) \ \ mit\ h(x_i) = w^d x + b \tag{3.6}$$

L ist die Loss-Funktion eines Modells (in unserem Fall die der logistischen Regression). Sie gibt Auskunft über die Anzahl an Missklassifikationen. R ist ein Regularisierungsterm, der die

Modellkomplexität bestraft, um ein Overfitting des Modells zu verhindern. Die Anpassung der Gewichtungen β_i der Entscheidungsfunktion basiert auf dem Gradientenabstiegsverfahren. Wie in Formel (3.7) gezeigt, wird anhand der partiellen Ableitungen des regularisierten Trainingsfehlers iterativ eine neue Gewichtung $w_k + 1$ berechnet. Wobei α_k die Lernrate, die Schrittweite im Parameterraum vorgibt. [62][38]

$$w_{k+1} = w_k \ - \ \alpha_k \nabla E(w_k, b), \tag{3.7}$$

Die Wahl der Lernrate α_k hat einen entscheidenden Einfluss darüber, ob eine Optimierungsaufgabe mittels SGD gelöst werden kann. Wird die Lernrate α_k zu klein gewählt, wird mit jeder Iteration der regularisierte Trainingsfehler minimiert. Der regularisierte Trainingsfehler konvergiert jedoch extrem langsam und es kann vorkommen, dass das Minimum nach Ablauf aller Iterationen noch nicht erreicht ist. Zudem droht die Gefahr, dass gerade bei mehrdimensionalen Funktionen die Zielfunktion gegen ein lokales Minimum konvergiert. In diesem Fall ist es nicht möglich, das globale Minimum zu finden und die Optimierungsaufgabe kann nicht sinnvoll gelöst werden. Durch die Wahl einer größeren Lernrate können lokale Minima überwunden werden. Ist die Lernrate zu groß, kann es jedoch passieren, dass die zu optimierende Funktion oszilliert oder divergiert. [62][38]

Bei einem herkömmlichen Gradientenabstiegsverfahren wird bei jeder Iteration der regularisierte Trainingsfehler für jede Instanz der Trainingsdaten berechnet. Dies bedeutet einen enormen Rechenaufwand. Um die benötigte Rechenleistung beziehungsweise Rechendauer zu reduzieren, wird beim SGD bei jeder Iteration der regularisierte Trainingsfehler nur für eine zufällige Auswahl an Trainingsinstanzen berechnet. Anhand dieser reduzierten Anzahl an Instanzen wird dann die neue Gewichtung w_{k+1} berechnet. [55]

3.3.4. Support Vector Machine

Die Support Vector Machine ist ein Klassifikationsverfahren für die binäre Klassifikation, kann jedoch unter anderem durch das Anwenden des One vs. Rest Verfahren auch für Klassifikationsaufgaben mit mehr als zwei Klassen eingesetzt werden. Sie verfolgt das Ziel, eine optimale Trennebene zwischen den Klassen zu finden. Diese sogenannte Hyperplane dient als lineare Entscheidungsgrenze, um eine Instanz seiner kategoriellen Zielgröße zuzuordnen. Diese Hyperplane 3.8 ist ein $(p-1)$ dimensionales Objekt im p dimensionalen Raum, wobei p die Anzahl an Merkmalen ist. [56][64]

$$h(x) = w^d x + b \tag{3.8}$$

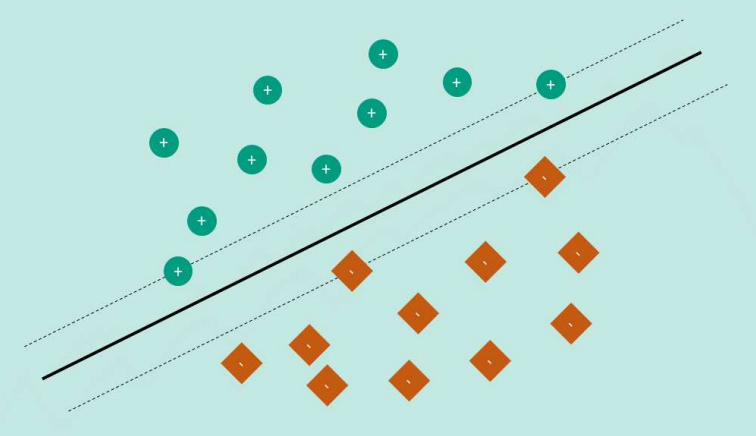


Abbildung 3.8.: Hyperplane einer Support Vector Machine mit maximalem Functional Margin (In Anlehnung an [64]).

Diese Hyperebene gilt als optimal, wenn der Functional Margin, also der Abstand zwischen den Instanzen der Trainingsdaten und der Hyperplane maximal ist. Mithilfe der Einführung einer Slack Variablen lassen sich Outlier, die eine lineare Trennlinie verhindern, kompensieren. Trotz dieser Anpassung ist in vielen Datensätzen keine zufriedenstellende lineare Trennung zwischen den Gruppen möglich. Mithilfe des Cover's Theorem kann dieses Problem gelöst werden. Werden die Daten des Datensatzes in einen höherdimensionalen Raum transformiert, erhöht sich damit nach dem Covers Theorem die Wahrscheinlichkeit einer linearen Trennung der Gruppen. Da die Dimensionserweiterung sehr rechenaufwendig ist und sich die benötigte Berechnungsdauer stark erhöhen würde, wird der sogenannte Kernel Trick eingesetzt. Dadurch können die Daten transformiert werden, ohne die Transformation komplett rechnerisch durchzuführen. Es wird eine Kernelfunktion verwendet, die im Merkmalsraum liegt, sich aber wie ein Skalarprodukt im höherdimensionalen Raum verhält. [58][56][64]

3.3.5. Künstliches neuronales Netz

Künstliche neuronale Netze können sowohl für das Supervised Learning als auch für das Unsupervised Learning verwendet werden. Die Funktion ist der von biologischen Nervensystemen nachempfunden. Ein neuronales Netz besteht aus vielen einzelnen Neuronen, die geschickt miteinander verschaltet sind. Diese kleinen Einheiten übernehmen die Informationsverarbeitung. Die in das Neuron j eingehenden Informationen o_i werden mithilfe einer Propagierungsfunktion zu einem Nettoeingabewert net_j verarbeitet. Als Propagierungsfunktion können verschiedene Formeln angewandt werden. Häufig wird eine Summenfunktion, bestehend aus Gewichtungen w_{ij} und den eingehenden Informationen o_i verwendet (3.9). [3]

$$net_j = \sum_{i=1}^{n} w_{ij} o_i \tag{3.9}$$

Ein Neuron besitzt dabei zwei Aktivierungszustände. Der Aktivierungszustand ist dabei abhängig von der Aktivierungsfunktion. Ist der Nettoeingabewert net_j größer als ein Schwellenwert σ_j, ist das Neuron aktiviert „1“. Ansonsten ist es inaktiv „0“. [3]

$$H(x) = \begin{cases} 1 & if\ net_j > \sigma_j \\ 0 & otherwise \end{cases} \tag{3.10}$$

Jedes Neuron übernimmt einen eigenständigen Rechenschritt. Anhand des Input, der aus dem gewichteten Output mehrere Neuronen der vorherigen Schicht besteht, wird ein Output bestimmt. Dieser beeinflusst wiederum, welche Neuronen der nachfolgenden Schicht aktiviert werden. Das Besondere bei neuronalen Netzen ist jedoch, dass die Informationsverarbeitung im Hidden Layer nicht definiert wird. Sie wird anhand eines selbstständigen Lernprozesses ermittelt. Mit jedem Lernschritt wird dabei versucht, den Zusammenhang zwischen Input und Output besser wiederzugeben. Dafür werden unter anderem die Verschaltung der Neuronen oder die Gewichtung der Eingabewerte angepasst. [3]

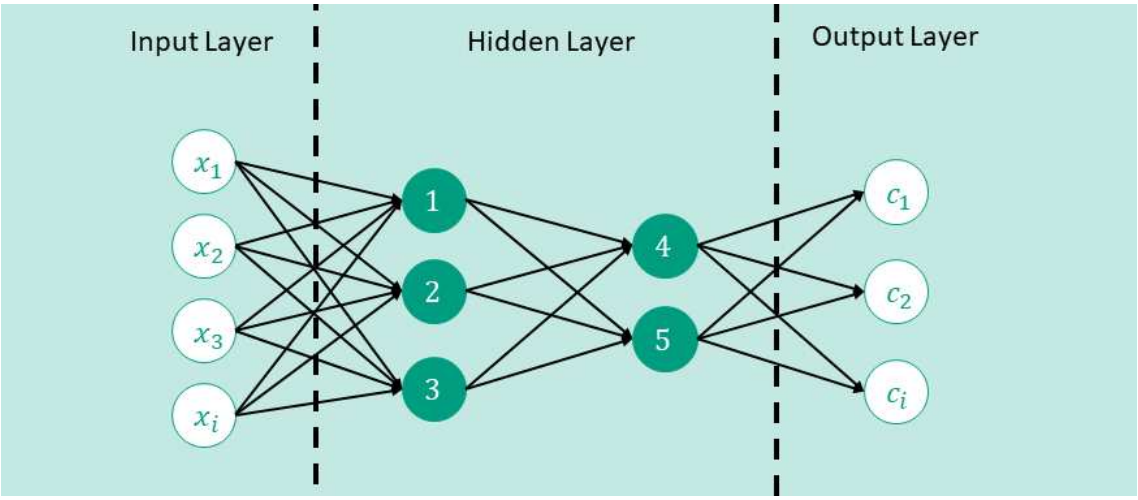


Abbildung 3.9.: Schichtenmodell eines neuronalen Netzes (In Anlehnung an [3]). Erste Schicht (links) ist das Input Layer gefolgt vom Hidden Layer (mitte). Letzte Schicht ist das Output Layer (rechts) .

Wie in Abbildung 3.9 zu erkennen, besteht ein künstliches neuronales Netz in den meisten Fällen aus drei aufeinander aufbauenden Schichten.

- Das **Input Layer** bildet alle Features der Instanzen als Eingabeneuronen wieder und bewirkt die Aktivierung der im Hidden Layer nachgeschalteten Neuronen. [3]
- Das **Hidden Layer** kann genau genommen aus mehreren Schichten an Neuronen bestehen. Input der Neuronen ist der Output der vorgeschalteten Neuronen. Abhängig von diesem Input wird ein Output mithilfe einer Aktivierungsfunktion, bestimmt. Dieser wird wiederum an nachgeschaltete Neuronen weitergegeben. [3]
- Abschließend existiert für jedes mögliche Klassenlabel ein Neuron. Diese Schicht wird auch **Output Layer** genannt. Der Output dieser Neuronen ist eine Wahrscheinlichkeit die Klassenzugehörigkeit. [3]

3.3.6. Decision Tree

Entscheidungsbäume (engl. Decision Trees) stellen ein nicht parametrisches Verfahren zur Klassifikation und Regression dar. Zur Klassenzuordnung wird dafür ein hierarisches Verfahren mit klar definierten Entscheidungsregeln entworfen. Aufgrund dieser klaren Entscheidungsregeln ist der Entscheidungsprozess für den Menschen im Vergleich zu anderen ML Modellen deutlich nachvollziehbarer [35]. Wie in Abbildung 3.10 zu sehen, kann der Entscheidungsprozess als ein gerichtetes Baumdiagramm dargestellt werden. Ein Entscheidungsbaum besteht aus einer Vielzahl von Knoten und Pfaden. An jedem Knoten, beginnend bei der Quelle des Entscheidungsbaums, dem sogenannten Wurzelknoten, erfolgt eine Abfrage an Features eines Objektes. In Abhängigkeit des Ergebnisses dieser Abfrage wird nun der entsprechende Ast gewählt, der zu einem weiteren Knoten führt. Dieser Vorgang wiederholt sich bis der gewählte Ast in einer Senke endet. Die Senke gibt Auskunft über das Klassenlabel, das dem Objekt zugeordnet werden soll. Formal betrachtet wird ein Objekt in Abhängigkeit seiner Attribute in jedem Knoten einem Subspace an möglichen Hypothesen zugeordnet, bis eine eindeutige Klassenzuweisung möglich ist. [9]

Im ML gibt es verschiedene Verfahren zur Generierung des Entscheidungsbaums, wobei alle nach dem Top Down Prinzip funktionieren. Sie führen also einen gerichteten Entscheidungsprozess durch. In dieser Thesis wird der CART Algorithmus, eine Abwandlung des C4.5 Algorithmus, eingesetzt. Beim CART Algorithmus handelt es sich um eine binäre Abfrage von Attributen. Der Entscheidungsbaum besitzt an jedem Knoten genau zwei Äste. Erfüllt das zu kategorisierende Objekt die abgefragten Attribute, wird der „$True$“ Pfad gewählt. Sind die Eigenschaften nicht erfüllt, wird „$False$“ gewählt. [46]

Bei aufwendigen Klassifikationsaufgaben neigt Decision Tree zum Overfitting. Dabei werden Modelle über komplexe Entscheidungsketten erzeugt, die zwar die Objekte der Trainingsdaten richtig zuordnen können, jedoch keine allgemeine Lösung zur Klassifikation der Daten

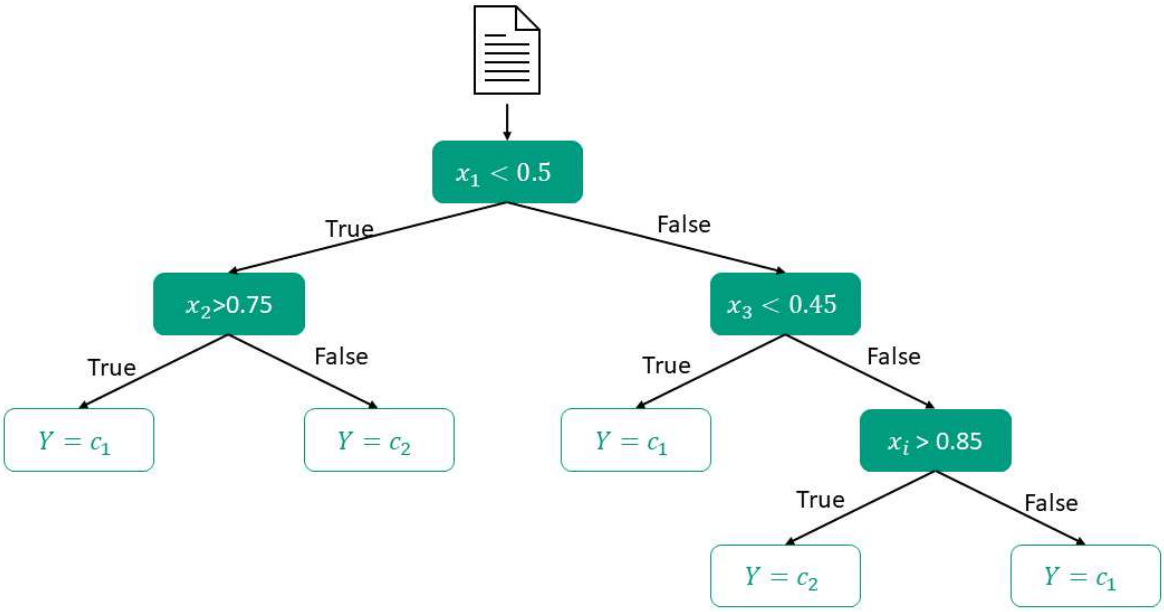


Abbildung 3.10.: Baumdiagramm eines Decision Tree (In Anlehnung an [9]). In Abhängigkeit der Features einer Instanz wird es in jedem Knoten einer Unterkategorie zugeordnet.

generieren kann. Um dieses Overfitting zu vermeiden, können Verfahren wie das Beschränken der Tiefe der Pfade eingesetzt werden. Ein weiterer Nachteil ist die große Varianz von Entscheidungsbäumen. Kleine Abweichungen in den Trainingsdaten können zu komplett unterschiedlichen Baumstrukturen führen. [9]

3.3.7. Random Forest

Random Forest ist ein Ensemble-Verfahren zur Klassifikation und Regression. Es ist eine Weiterentwicklung des Decision Tree, um die zuvor beschriebenen Schwächen abzumildern. Beim Random Forest Verfahren wird ein diverses Set an k Entscheidungsbäumen erzeugt. Die Diversität der Entscheidungsbäume ist zum einen auf das Training mittels Bootsstrap Aggregation zurückzuführen. Bei der Bootsstrap Aggregation wird der zum Training verwendetete Datensatz S in k zufällige Stichproben aufgeteilt, wobei die relative Häufigkeit der Klassenlabels in jeder Stichprobe erhalten bleibt. Anhand dieser k Stichproben werden k voneinander unabhängige Entscheidungsbäume erzeugt. Um die Diversität der Entscheidungsbäume zu steigern, wird zudem für jede Stichprobe eine reduzierte, zufällige Auswahl an Features verwendet. Soll eine Instanz X mittels Random Forest klassifiziert werden, trifft

zunächst jeder der k Decision Trees eine eigenständige Vorhersage. Die Vorhersagen werden anschließend durch einen Mehrheitsentscheid kombiniert. Das Klassenlabel, das am häufigsten vorhergesagt wurde, wird als finale Vorhersage gewählt. Der Zusammenschluss einer Vielzahl an Decision Trees kann den Verallgemeinerungsfehler verbessern und reduziert damit die Gefahr des Overfitting eines einzelnen Decision Tree. [8][12]

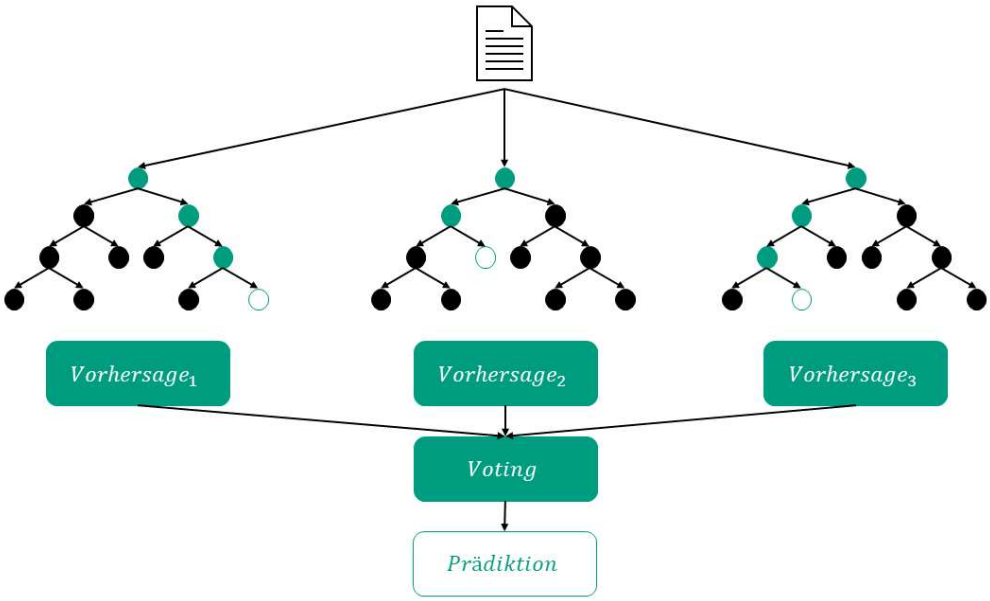


Abbildung 3.11.: Funktionsschema eines Random Forest Klassifikators (In Anlehnung an [12]). Die Vorhersagen unabhängiger Entscheidungsbäume werden mittels Mehrheitsentscheid kombiniert.

3.4. Diversität der Klassifikatoren

Für die erfolgreiche Anwendung von Ensemble-Verfahren ist ein entscheidender Faktor die Diversität der zu kombinierenden Klassifikatoren. Diversität der Klassifikatoren kann unter anderem durch das Verwenden unterschiedlicher Lernalgorithmen, anderer Featurauswahl oder durch das Training anhand von unterschiedlichen Daten erzeugt werden. [31][64]

Für eine erfolgreiche Kombination der Vorhersagen der Basisklassifikatoren müssen die Basisklassifikatoren voneinander abweichende Vorhersagen machen. Es wird also eine Diversität

der Prädiktionen benötigt. Die Vorhersagen der Basisklassifikatoren sollten so genau wie möglich sein. Kommt es zu fehlerhaften Hypothesen, sollten die Klassifikatoren unterschiedliche Instanzen falsch klassifizieren. Die Korrelation der fehlerhaften Hypothesen gibt also Auskunft darüber, ob es sich um ein diverses Set an Klassifikationen handelt. [31][64]

Zur Ermittlung der Diversität wird in dieser Thesis die paarweise Abweichung von Klassifikatoren ermittelt. Anhand einer Kontingenztafel kann der Anteil an unterschiedlichen fehlerhaften Prädiktionen dargestellt werden. Abbildung 3.12 zeigt den Aufbau einer Kontingenztafel für eine binäre Klassifikationsaufgabe. a sind die sowohl von $h_i(x)$ und $h_j(x)$ als auch 1 klassifizierten Instanzen. b, die von beiden als 0 Klassifizierten und b und c wurden von den Klassifikatoren unterschiedlich klassifiziert. Zudem wird eine Variable $m = a + b + c + d$ eingeführt. Sie umfasst die Summe aller Vorhersagen beider Klassifikatoren. [64]

		h_j 1	0
h_i	1	a	b
	0	c	d

Abbildung 3.12.: Kontingenztafel zur Berechnung der Diversität (In Anlehnung an [64]).

Anhand dieser Kontingenztafel kann zudem das von Skalak eingeführte **Disagreement Measure** dis_{iJ} ermittelt werden [44]. Das Disagreement Measure hat einen Wertebereich von [0,1]. Je größer der ermittelte Wert, desto größer ist die Diversität zwischen zwei Klassifikatoren. [31][64]

$$d_{ij} = \frac{a + d}{a + b + c + d} \tag{3.11}$$

Für Klassifikationsaufgaben mit mehr als zwei Klassen wird die Kontingenztafel angepasst. Dabei umfasst a die Anzahl an Instanzen, die von beiden Klassifikatoren korrekt klassifiziert wurden und d wurde von beiden falsch klassifiziert. b und c gibt an, wie viele Instanzen jeweils nur von einem Klassifikator korrekt klassifiziert werden konnten. Bei Klassifikationsaufgaben, bei denen beide Klassifikatoren eine sehr hohe Genauigkeit aufweisen, wird das Disagreement Measure gegen null gehen. Um dennoch eine Auskunft über die Diversität zu erhalten, kann es sinnvoll sein, m um a zu bereinigen. Der so ermittelte Wert gibt nun Auskunft über die Diversität der falsch klassifizierten Objekte.

Für alle verwendeten Datensätze wurde anhand der Kontingenztabelle und dem Disagreement Measure die Diversität der Klassifikatoren untersucht. Dabei konnte für alle Datensätze Diversität in den Prädiktionen der Basisklassifikatoren festgestellt werden. Tabelle 3.2 zeigt

Tabelle 3.2.: Disagreement Measure der Basisklassifikatoren des MNIST Digitsdatensatz in [%]. Das Disagreement Measure wurde um Instanzen bereinigt, die von beiden Klassifikatoren korrekt klassifiziert wurden.

Klassifikator	LG	SGD	DTREE	Forrest	CNN	SVM	LDA
LG	0	36,1	74,96	60,67	41,1	35,31	60,43
SGD	36,1	0	74,16	54,78	39,9	26,89	54,77
DTREE	74,96	74,16	0	71,55	72,45	73,8	74,73
Forrest	60,67	54,78	71,55	0	54,91	58,5	62,23
CNN	41,1	39,9	72,45	54,91	0	41,9	54,81
SVM	35,31	26,89	73,8	58,5	41,9	0	53,58
LDA	60,43	54,77	74,73	62,23	54,81	53,58	0

die um korrekte Prädiktionen bereinigten Disagreement Measures für die Basisklassifikatoren des MNIST Digits Datensatzes. Dabei ist das Disagreement Measure in Prozent angegeben.

3.5. Voting

Zur Konsensfindung wird in dieser Arbeit ein Pluralitäts Softvoting angewandt. Jeder der sieben Basisklassifikatoren gibt als Output für jedes Label eine A-posteriori Wahrscheinlichkeit aus. Als finale Vorhersage wird das Label gewählt, welches den größten Gesamtstimmanteil aller Basisklassifikatoren bekommt. Da die Klassifikationsleistung der Basisklassifikatoren sich teils stark unterscheidet, erweist es sich als sinnvoll, bei der Konsensbildung die Vorhersagen der Klassifikatoren in Abhängigkeit ihrer Stärke zu gewichten. Somit haben Basisklassifikatoren mit einer hohen Güte stärkeren Einfluss auf das Voting als schwache Klassifikatoren. Ist die relative Häufigkeit der einzelnen Klassenlabels nahezu gleichverteilt (MNIST Digits & Fashion MNIST) wird die erreichte Accuracy der Basisklassifikatoren für die Validationsdaten als Gewichtung verwendet. [31][64]

$$H(x) = c_{\underset{argmax}{} \sum_{I=1}^{T} accuracy_i * h_i^k(x)} \tag{3.12}$$

In der Realität ist die Häufigkeitsverteilung der Klassenlabels oft sehr unausgeglichen. Ein Klassifikator kann eine hohe Accuracy besitzen, obwohl dieser die Klassifikationsaufgabe nicht zufriedenstellend lösen kann. Für solche Klassifikationsaufgaben sollte deshalb ein

anderer Parameter als Gewichtung gewählt werden. Demzufolge wird für die Datensätze der MedMNIST Datenbank eine andere Gewichtung gewählt. Es wird davon ausgegangen, dass ein guter Klassifikator für diese Klassifikationsaufgaben sowohl eine hohe Precision als auch Recall aufweisen sollte. Deshalb wird als Gewichtung der F1-Score gewählt. [31][64]

$$H(x) = c_{\underset{argmax}{} \sum_{I=1}^{T} f_1 - measure * h_i^k(x)} \tag{3.13}$$

3.6. Blending

Blending ist ein Ensemble-Verfahren, bei dem die Vorhersagen der Basisklassifikatoren mithilfe eines Meta-Klassifikators kombiniert werden. Dieser Meta-Learner erlernt anhand der Vorhersagen der Basisklassifikatoren für den Validationsdatensatz ein Schemata, wie die Vorhersagen der Basisklassifikatoren kombiniert werden sollen. Essenziell für eine verbesserte Klassifikationsleistung ist neben der Genauigkeit und der Diversität der Basisklassifikatoren die Wahl eines geeigneten Meta-Klassifikators [17]. In dieser Arbeit werden zwei unterschiedliche Algorithmen als Meta-Learner untersucht: Decision Tree und logistische Regression. Neben der Wahl des Meta-Klassifikators ist auch die Zusammensetzung der Metadaten ein wichtiger Faktor für eine erfolgreiche Kombination der Vorhersagen. So können Parameter wie die Entropie Auskunft über die Konfidenz eines Klassifikators geben. [53][18]

Logistische Regression

Wird die logistische Regression für das Meta-Learning verwendet, wird die Klassifizierungsaufgabe in K binäre Aufgaben aufgeteilt, wobei k für die Anzahl an Klassen steht. Jeder der binären Klassifikatoren h_i gibt Auskunft über die A-posteriori Wahrscheinlichkeit, ob ein Objekt zur Klasse C_i gehört oder nicht. Für jede der K Klassifikationsaufgaben wird eine Diskriminanzfunktion der Form

$$z(x) = \beta_o + \beta_1 x_1 + ... + \beta_j x_j \tag{3.14}$$

gebildet. Wobei $x_i, .., x_j$ die Metafeatures, also die Vorhersagen und die daraus abgeleiteten Parameter der Basisklassifikatoren sind. Die Gewichtungen $_i$ werden dabei durch ein iteratives Verfahren so geschätzt, dass die Loss-Funktion für die Vorhersagen der Validationsdaten minimal ist. Die Diskriminanzfunktion wird anschließend mithilfe der in Kapitel 3.3.1 vorgestellten Funktion der logistischen Regression in den Wertebereich $[0, 1]$ transformiert. Das Ergebnis der Transformation stellt dabei die A-posteriori Wahrscheinlichkeit eines Objektes x_i zur Klasse C_i dar. Für alle der k Klassifikationsaufgaben wird diese Wahrscheinlichkeit berechnet. Wie in

Formel 3.15 dargestellt, wird das Klassenlabel mit der größten A-posteriori Wahrscheinlichkeit anschließend als Prädiktion des Blending-Verfahrens gewählt. [2][53]

$$H(x) = Y_k \; wenn \; p_k > p_i \; mit \; p_i \in \{p_1, .., p_k\} \tag{3.15}$$

Für den MNIST Digits Datensatz mit 10 Klassenlabels werden also zehn Diskriminanzfunktionen erstellt. Basierend auf den Prädiktionen der Basisklassifikatoren wird anhand der Diskriminanzfunktionen und der Funktion der logistischen Regression nun die A-posteriori Wahrscheinlichkeiten ermittelt. Ist für eine Instanz x_i mit den Metafeatures $h_1(x_i), ...h_7(x_i)$ nun die A-posteriori Wahrscheinlichkeit $p_C(c_k|x_i)$ maximal, wird das Objekt der Klasse c_k zugeordnet.

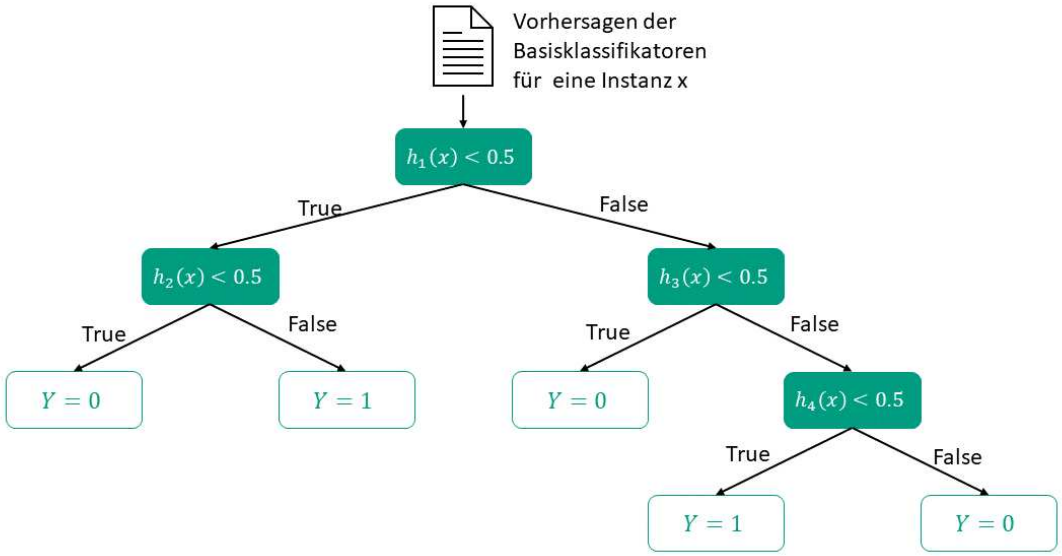


Abbildung 3.13.: Meta Decision Tree (In Anlehnung an [9]). In Anhängigkeit der Vorhersagen der Basisklassifikatoren wird eine Instanz in jedem Knoten einer Unterkategorie zugeordnet.

Decision Tree

Beim Verwenden des Decision Tree als Meta-Learner wird anhand der Meta-Features, also den Vorhersagen der Klassifikatoren und den daraus abgeleiteten Parametern ein gerichteter Entscheidungsbaum erzeugt. Da der CART Algorithmus verwendet wurde, teilt sich der Baum an jedem Knoten in zwei Folgeknoten auf. An jedem Knoten wird ein Meta-Feature abgefragt. In Abhängigkeit davon, ob die abgefragte Eigenschaft erfüllt ist, wird nun einer der beiden ausgehenden Äste gewählt. Die Instanz wird also einer Teilmenge zugeordnet. Dieser Vorgang

wiederholt sich in den darauffolgenden Knoten so lange, bis der gewählte Pfad in einer Senke endet. Die Senken enthalten die Informationen über die Klassenzugehörigkeit. Allen Instanzen, die der Teilmenge der gewählten Senke zugeordnet wurden, wird das gleiche Klassenlabel zugewiesen. In Abbildung 3.13 ist beispielhaft das Entscheidungsdiagramm für eine binäre Klassifikationsaufgabe zu sehen. Meta-Features sind dabei die diskreten Vorhersagen der Basisklassifikatoren. In jedem Knoten wird eine Instanz X, in Abhängigkeit der Vorhersage eines Basisklassifikators, einer Teilmenge zugeordnet. Dieses Vorgehen wird solange wiederholt, bis die Objekte einer Teilmenge alle derselben Klasse angehören. [53][52] [9]

Meta-Features

Wie zuvor schon erwähnt, ist auch die Auswahl an Features der Metadaten entscheidend, ob eine Verbesserung der Klassifikationsleistung erreicht werden kann. In dieser Arbeit werden sowohl für das Blending mit Decision Tree als auch mit logistischer Regression als Meta-Learner folgende Meta-Features implementiert:

- **Class Probabilility**: $h_i(x) = [0, 1]$ Output der Basisklassifikatoren ist die Wahrscheinlichkeitsverteilung der Klassenlabels. Die Meta-Features bestehen also aus einem $7 * k$ dimensionalen Vektor. Sie enthalten die Wahrscheinlichkeitsverteilungen für die Vorhersage der Basisklassifikatoren. Jede Wahrscheinlichkeitsverteilung besteht aus k Features, wobei k für die Anzahl an möglichen Klassenlabels steht.
- **argmax**: Als Meta-Features werden die diskreten Vorhersagen der Basisklassifikatoren verwendet. Also die Klassenlabels mit der größten Class Probability [53]. Die Meta-Features bestehen aus einem 7-dimensionalen Vektor, wobei jede Dimension die Vorhersage eines Klassifikators enthält.

$$maxprob(x, C) = max\ p_c(c_i|x) \tag{3.16}$$

- **argmax, Entropie**: $h_i(x) = [0, 1]$ Die Meta-Features bestehen wieder aus den Vorhersagen der Basisklassifikatoren. Als zusätzliche Attribute werden die Entropien der Class Probability der Klassifikatoren verwendet. Die Entropie kann dabei als Maß für die Konfidenz einer Hypothese eines Basisklassifikators gesehen werden und wird mit Formel 3.17 berechnet [53]. Geht der Wert der Entropie gegen null, ist die Entropie gering, bei eins ist sie hoch. Die Meta-Features bestehen also aus einem 14 dimensionalen Vektor, wobei in sieben Dimensionen die Vorhersagen der Klassifikatoren gespeichert sind. Die restlichen 7 Dimensionen enthalten die Entropien der Vorhersagen der Klassifikatoren. [53]

$$Entropie(x, C) = -\sum p_C(c_i|x) * log_2 p_C(c_i|x) \tag{3.17}$$

Insgesamt werden sechs Blending-Verfahren implementiert, die sich entweder in der Wahl des Meta-Learner oder in der Zusammensetzung der Meta-Features unterscheiden. Im Folgenden sind die unterschiedlichen Implementierungen aufgelistet:

- Meta Decision Tree (MDT) verwendet einen Decision Tree als Meta-Layer. Die Metadaten bestehen aus den Class Probabilties. Die Meta-Features bestehen aus einem $k * 7$ dimensionalen Vektor.
- Meta Decision Tree2 (MDT2) verwendet einen Decision Tree als Meta-Layer. Die Metadaten sind die diskreten Vorhersagen der Basisklassifikatoren. Die Meta-Features bestehen aus einem 7 dimensionalen Vektor.
- Meta Decision Tree3 (MDT3) verwendet einen Decision Tree als Meta-Layer. Die Metadaten bestehen aus den diskreten Vorhersagen der Basisklassifikatoren und den Entropien der Class Probabilities.Die Meta-Features bestehen aus einem $k * 2$ dimensionalen Vektor.
- Meta Logistische Regression (MLR) verwendet die logistische Regression als Meta-Layer. Die Metadaten bestehen aus den Class Probabilties. Die Meta-Features bestehen aus einem $k * 7$ dimensionalen Vektor.
- Meta Logistische Regression2 (MLR2) verwendet die logistische Regression als Meta-Layer. Die Metadaten sind die diskreten Vorhersagen der Basisklassifikatoren. Die Meta-Features bestehen aus einem k dimensionalen Vektor.
- Meta Logistische Regression3 (MLR3) verwendet die logistische Regression als Meta-Layer Die Metadaten bestehen aus den diskreten Vorhersagen der Basisklassifikatoren und den Entropien der Class Probabilities.Die Meta-Features bestehen aus einem $k * 2$ dimensionalen Vektor.

4. Methoden zur Evaluierung der Klassifikatoren

In diesem Kapitel werden die Verfahren zur Evaluierung der Klassifikationsleistung der Ensemble Methoden vorgestellt. Um die Zuverlässigkeit der Klassifikatoren zu bewerten, werden für die Prädiktionen der Testdaten Parameter wie die Accuracy und der Recall erhoben. Anschließend werden die unterschiedlichen Klassifikationsverfahren anhand dieser Metriken miteinander verglichen. Zur Evaluierung der Robustheit werden mithilfe von Fault Injections unterschiedliche Test Sets implementiert. Die einzelnen Instanzen der Testdaten enthalten dabei Pertubationen in den Features. Anhand des Klassifikationsergebnisses der wird nun die Fehleranfälligkeit der Systeme bei Störungen untersucht.

4.1. Bewertung von Klassifikatoren

Um eine quantitative Aussage über die Zuverlässigkeit eines Klassifikators treffen zu können, werden Parameter benötigt, anhand derer diese erfasst werden kann. Unter der Annahme, dass jedes Element eindeutig einer Klasse angehört, gibt es bei der binären Klassifikation grundsätzlich vier verschiedene Zustände, die eine Hypothese bezüglich ihres Wahrheitsgehaltes darstellen können:

- **True Positive**(TP): Die Vorhersage und der tatsächliche Wert sind identisch: ein Element wurde als wahr vorhergesagt und ist tatsächlich wahr [30].
- **True Negative**(TN): Die Vorhersage und der tatsächliche Wert sind identisch: ein Element wurde als falsch vorhergesagt und ist tatsächlich falsch [30].
- **False Positive**(FP): Die Vorhersage und der tatsächliche Wert sind nicht identisch: ein Element wurde als wahr vorhergesagt, aber ist tatsächlich falsch. Es wird auch vom Fehler Typ 1 gesprochen [30].

- **False Negative**(FN): Die Vorhersage und der tatsächliche Wert sind nicht identisch: ein Element wurde als falsch vorhergesagt, aber ist tatsächlich wahr. Es wird auch vom Fehler Typ 2 gesprochen [30].

Dabei stellen die Ereignisse True Positive und False Positive eine richtige Vorhersage und die Ereignisse False Positive und False Negative eine fehlerhafte Prädiktion dar. Die relativen Häufigkeiten der Wahrheitszustände können, wie in Abbildung 4.1, als Konfusionsmatrix dargestellt werden. [30]

		Klassenlabel: c_i		
		1	0	Gesamt:
Prädiktion: $h(x)$	1	true positive (TP)	false positive (FP)	$\sum Prädiktion = 1$
	0	false negative (FN)	true negative (TN)	$\sum Prädiktion = 0$
	Gesamt:	$\sum Klassenlabel = 1$	$\sum Klassenlabel = 0$	

Abbildung 4.1.: Konfusionmatrix einer binären Klassifikationsaufgabe (In Anlehnung an [30]).

Anhand der Konfusionsmatrix können Kenngrößen zur Beurteilung der Zuverlässigkeit des Klassifikators abgeleitet werden.

Accuracy

Die Accuracy bestimmt die Genauigkeit eines Modells, also das Verhältnis von korrekt klassifizierten Hypothesen zu der Summe aller Hypothesen. [30]

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

Da sich die relative Häufigkeit der verschiedenen Klassenlabels oft stark unterscheidet, ist dieser Parameter nur bedingt aussagekräftig. So kann es vorkommen, dass ein Klassifikationsmodell eine hohe Genauigkeit aufweist und dennoch das zu lösende Problem nur ungenügend abbilden kann. Ein anschauliches Beispiel dafür wäre eine KI zur Detektion von malignen Tumoren. Es wird angenommen, dass ungefähr 10 Prozent der Probanden an einem malignen Tumor erkranken. Die Vorhersage ist positiv, wenn der Proband erkrankt ist und negativ,

wenn er gesund ist. Die entwickelte KI ist jedoch fehlerhaft und sagt für jeden Probanden eine negative Diagnose vorher. Dennoch hätten wir so eine Accuracy von 90 Prozent. Aus diesem Grund gibt es noch weitere Parameter, die berücksichtigt werden müssen, um eine valide Aussage über die Zuverlässigkeit eines Klassifikators treffen zu können. [30]

Precision

Der positive Vorhersagewert (Precision) ist das Verhältnis von korrekt als positiv klassifizierten Hypothesen zu der Summe aller positiv zu klassifizierenden Hypothesen. Sie gibt an, wie viel Prozent aller als positiv zu klassifizierenden Objekte erfolgreich als positiv klassifiziert wurden. Eine Präzision von 1 bedeutet, dass alle als positiv zu klassifizierten Objekte auch erfolgreich positiv klassifiziert wurden. Diese Metrik ist besonders relevant, wenn ein fälschlicherweise positiv klassifiziertes Objekt hohe Kosten verursacht. [30]

$$precision = \frac{TP}{TP + FP} \tag{4.2}$$

Recall

Die Sensitivität (Recall) beschreibt die Trefferquote, also das Verhältnis von korrekt positiv klassifizierten Hypothesen zu der Summe aller negativen Hypothesen. Sie gibt also die Wahrscheinlichkeit an, dass ein positives Objekt auch tatsächlich als positiv klassifiziert wird. Ein Recall von 1 gibt an, dass alle als positiv zu klassifizierenden Objekte vom Modell auch als positiv erkannt wurden. [30]

$$recall = \frac{TP}{TP + FN} \tag{4.3}$$

$\mathbf{F}_1 Measure$

Oftmals sind eine hohe Precision und und ein hoher Recall für die Güte eines Klassifikators entscheidend. Zur vereinfachten Bewertung dieser beiden Metriken wurde das F_1 measure, häufig auch F_1 Score genannt, eingeführt. Es ist das harmonische Mittel von Recall und Precision. Der Indizes “1„ bedeutet, dass Recall und Precision gleich gewichtet werden. [30]

$$F_1\ Measure = 2 * \frac{precision * recall}{precision + recall} \tag{4.4}$$

Multi-Class-Problem

Für Klassifikationsaufgaben mit $K > 2$ Labeln muss die Berechnung dieser Metriken angepasst werden. In dieser Thesis wird davon ausgegangen, dass die korrekte Klassifikation der unterschiedlichen Labels gleich wichtig ist. Es erfolgt also keine Anpassung an die relative Häufigkeit der Klassen. Man spricht auch vom Macro Averaging. Die Klassifikationsergebnisse werden in k binäre Probleme aufgeteilt. Nach Formel 4.2 und 4.3 werden Precision, Recall und F1-Score berechnet. Anschließend werden die Ergebnisse gemittelt. Die Berechnung der Accuracy kann Formel 4.5 entnommen werden. [50]

$$accuracy = \frac{Anzahl\ an\ korrekten\ Vorhersagen}{Anzahl\ aller\ Vorhersagen} \tag{4.5}$$

4.2. Fault Injections

In der Softwareentwicklung stellen Fehler Injektionen (eng. Fault Injections) eine in der Branche etablierte Technik dar, um die Robustheit einer Software und den Umgang mit Fehlern zu bewerten. Dabei werden systematisch Störungen in das System eingebracht und die resultierenden Auswirkungen untersucht [1]. Eine Störung stellt dabei eine unzulässige Abweichung von mindestens einer Eigenschaft des Systems gegenüber seinem zu erwartenden Verhalten dar. Das Auftreten solch einer Störung kann zu einem Ausfall oder einer Fehlfunktion der Anwendung führen. Fehlerquellen können dabei grundlegend in zwei Fehlerarten aufgeteilt werden: [32][29][5]

- **Softwarefehler**: Softwarefehler sind auf Designfehler in der Entwicklungsphase der Anwendung zurückzuführen. Der Fehler bleibt dabei solange im Programmcode der Software verborgen, bis der defekte Bereich aufgerufen wird. [29]
- **Hardwarefehler**: Hardwarefehler entstehen durch defekte physische Komponenten des Systems. Je nach Fehlerursache kann die Dauer der auftretenden Störung dabei permanent, intermittierend oder transient sein. Permanente und intermittierende Fehler sind dabei häufig auf defekte Komponenten des Systems zurückzuführen. Transiente Fehler entstehen stattdessen häufig aufgrund externer Störeinflüsse. So führen zum Beispiel extreme Umwelteinflüsse wie Hitze oder Strahlung zur temporären Störung von Sensoren oder Aktoren. [29]

In dieser Arbeit werden zur Evaluation der Robustheit dabei nur Fault Injections genutzt, die Störungen in den Inputdaten der Klassifikatoren simulieren. Die Stärke der Störgröße wird dabei iterativ erhöht, wobei in jedem Schritt der Einfluss auf das Klassifikationsergebnis untersucht wird. Es werden insgesamt vier verschiedene Fault Injections untersucht, wobei die

genaue Fehlerherkunft für diese Arbeit nicht weiter relevant ist. Beim Ursprung der Störung kann grundlegend zwischen natürlichen und vorsätzlichen Datenfehler unterschieden werden.

- **natürliche Datenfehler**: Dabei entsteht eine Korruption der Daten durch willkürliche auftretende Störungen. Diese Störungen führen zu Pertubationen in den Daten, die wiederum den Klassifikationsprozess beeinflussen. Häufig auftretende Pertubationen sind dabei Phänomene wie Unschärfe oder Rauschen. [36]
- **vorsätzliche Datenfehler**: Ein Angreifer versucht dabei vorsätzlich die Daten zu manipulieren. Mit einem mutwilligen Angriff auf das System wird dabei eine Pertubation in den Daten erzeugt, die oft für das menschliche Auge nicht zu erkennen ist, aber zu einer fehlerhaften Prädiktion der Anwendung führt. Dieser gezielte Angriff wird auch Adversarial Attack genannt. [36]

4.2.1. Salt & Pepper Noise

Unter Salz und Pfeffer Rauschen (eng. Salt & Pepper Noise) wird in der Bildverarbeitung das Phänomen von zufällig weiß oder schwarz gefärbten Pixeln verstanden. Dabei handelt es sich um eine spezielle Form des Impulsrauschens. Externe Einflüsse führen dazu, dass einzelne Pixel die Informationen über den gespeicherten Farbwert verlieren. Die betroffenen Pixel werden stattdessen zufällig in zwei Extremwerte digitalisiert. Die Pixel nehmen also entweder die minimale (0) oder maximale Intensität (255) an. Bei einer maximalen Intensität wird der Pixel schwarz dargestellt und bei einer minimalen Intensität weiß. Der Farbwert der Pixel, die nicht von der Störung betroffen sind, bleibt unverändert. Das Auftreten dieses Rauschens verschlechtert die Bildqualität, was für die Anwendung einen Verlust an Informationen über das zu klassifizierende Objekt bedeutet. Fehlen nun entscheidende Merkmale eines Objektes, kann es zu einer Fehlklassifikation kommen. [54][65]

Das Auftreten von Salt & Pepper Noise ist dabei in digitalen Bildern häufig auf Fehler in der Bildaufnahme und/oder -verarbeitung zurückzuführen. Häufige Ursachen sind dabei defekte Kamerasensoren oder Soft- und Hardwarefehler. Allgemein wird Impulsrauschen durch das Auftreten diskontinuierlicher Impulse oder durch Rauschspitzen erzeugt. Diese haben eine kurze Dauer, eine hohe Spektraldichte und eine große Amplitude. Eine mögliche Ursache für das Auftreten eines Impulsrauschens könnten dabei unter anderem elektromagnetische Impulse darstellen. [54][65]

In dieser Arbeit wird der Einfluss von Salt & Pepper Rauschen auf den Klassifikationsprozess untersucht. Die Störung kann nachgeahmt werden, indem zuerst die Dimension des Bildes ermittelt wird. Anschließend werden zufällig die Koordinaten zweier Pixel ausgewählt, wobei der Wert eines Pixels auf 0 und der des anderen auf 255 gesetzt wird. Um die Robustheit der Klassifikatoren gegenüber dem Salt & Pepper Noise zu untersuchen, wird iterativ die Anzahl

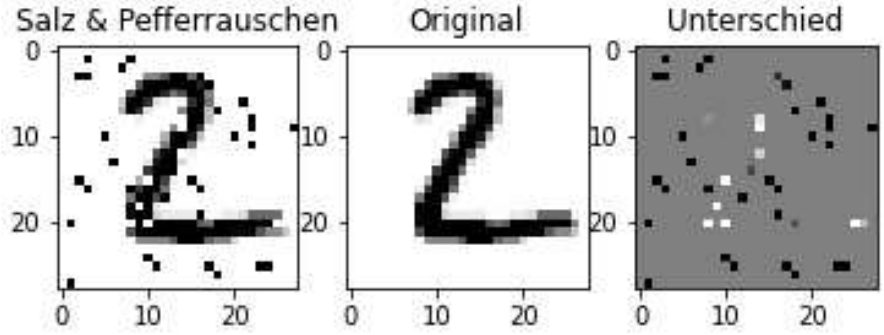


Abbildung 4.2.: Salt & Pepper Noise bei einer Instanz des MNIST Digits Datensatzes. (a) zeigt eine Instanz die durch Salt & Pepper Noise gestört ist, (b) zeigt das Originalbild und (c) zeigt die durch Salt & Pepper Noise entstandene Pertubation.

an manipulierten Pixeln erhöht und jedes Mal die in Abschnitt 4.1 bestimmten Metriken zur Evaluierung der Vorhersagen berechnet. Es werden insgesamt 20 Iterationen durchgeführt. In jedem Zyklus werden dabei zufällig je sechs Pixel weiß und schwarz gefärbt. Nach der letzten Iteration sind damit bis zu 30 % der Bildinformationen durch den Störprozess verloren gegangen.

4.2.2. Gaussian Noise

Bei der visuellen Datenerfassung ist das Auftreten von Rauschprozessen oft unvermeidbar und tritt in Form der Variation der Helligkeit oder Farbe eines Bildes auf. Viele der möglichen Störprozesse können anhand eines gaußverteilten Rauschens nachgeahmt werden. So können das thermische Rauschen oder das verstärkende Rauschen eines Bildsensors durch ein additives weißes Gauß Rauschen (AWGN) simuliert werden. Es wird dabei von einem weißen Gauss Rauschen ausgegangen, da anzunehmen ist, dass der Störprozess unkorreliert mit dem zu erfassenden Objekt ist. [33][14]

$$A = A_o + N \tag{4.6}$$

Abbildung 4.3 illustriert den Einfluss eines AWGN auf eine Instanz des MNIST Digits Datensatz dargestellt. Das erste Bild zeigt dabei das beobachtete Bild. Wie in Formel 4.6 beschrieben, setzt sich das beobachtete Bild A aus dem wahren Bild A_0 und einem signalunabhängigen Rauschprozess N zusammen. Die Häufigkeit der unterschiedlichen Störamplituden eines AWGN ist dabei gaußverteilt [57]. Mathematisch wird die Amplitude des Rauschens wie folgt berechnet:

$$p_G(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-\mu)^2}{2\sigma^2}} \tag{4.7}$$

Dabei ist σ die Standardabweichung des Rauschens und μ der Mittelwert des Rauschensignals. Soll einem Bild ein AWGN hinzugefügt werden, muss zunächst die mittlere Intensität und Standardabweichung festgelegt werden. Anschließend wird dem Bild eine unkorrelierte, normalverteilte Rauschstörung hinzugefügt [33][57]. In dieser Arbeit wird davon ausgegangen, dass die mittlere Intensität des Gauss Rauschen $\mu = 5$ ist. Zur Untersuchung des Einflusses eines AWGN auf den Klassifikationsprozess werden dabei Störprozesse mit einer unterschiedlichen Standardabweichung untersucht. Um die Robustheit der Klassifikatoren gegenüber dem Gauss Noise zu untersuchen, wird iterativ die Standardabweichung des Rauschens erhöht und jedes Mal die in Abschnitt 4.1 bestimmten Metriken zur Evaluierung der Vorhersagen berechnet. Es werden insgesamt 20 Iterationen durchgeführt. In jedem Zyklus wird die Standardabweichung des Rauschens um 5 erhöht. Nach der letzten Iteration haben die Inputdaten ein Gauss Rauschen mit einer Standartabweichung von 100.

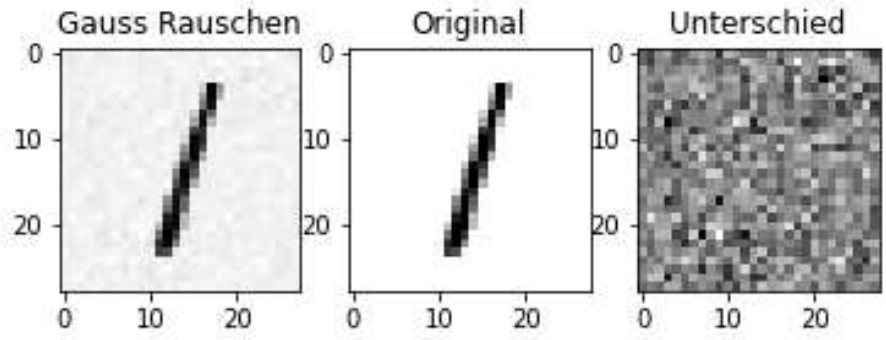


Abbildung 4.3.: Gaussian Noise bei einer Instanz des MNIST Digits Datensatzes. (a) zeigt eine Instanz, die durch Gaussian Noise gestört ist, (b) zeigt das Originalbild und (c) zeigt die durch Gaussian Noise entstandene Pertubation.

4.2.3. Gaussian Blur

Aufgrund von technischen Fehlern in Blende oder Fokus des Bildsensors, aber auch durch Bewegung des Messobjektes oder Messgeräts kann es vorkommen, dass die abzubildenden Objekte unscharf dargestellt werden [14]. Falls diese Störung nicht durch den Einsatz von Filtern behoben wird, kann dies den Klassifikationsprozess beeinträchtigen. Mithilfe eines Gaussian Blur Filters auf den Eingangsdaten sollen unscharfe Bildaufnahmen simuliert werden.

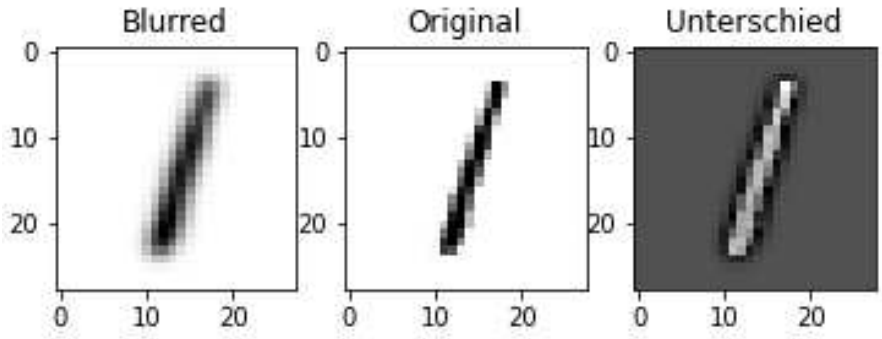


Abbildung 4.4.: Gaussian Blur bei einer Instanz des MNIST Digits Datensatzes. (a) zeigt eine Instanz die durch Gaussian Blur gestört ist, (b) zeigt das Originalbild und (c) zeigt die durch Gaussian Blur entstandene Pertubation.

In der Bildverarbeitung bezeichnet Gaussian Blur das Weichzeichnen eines Bildes mithilfe der Gauss Funktion. Der Filter glättet die Kanten eines Bildes, dadurch gehen kleine Bildstrukturen verloren. Markante Strukturen hingegen bleiben erhalten. Häufig werden Gaussian Filter zur Rauschreduzierung eines Bildes eingesetzt [23]. In dieser Arbeit wird der Filter jedoch mit dem Ziel eingesetzt, zu klassifizierende Objekte unscharf darzustellen. Wird der Filter auf ein Bild angewendet, werden die neuen Farbwerte anhand folgender Formel berechnet:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.8}$$

x und y sind dabei die Koordinaten, dessen Farbwerte neu bestimmt werden sollen. Die Varianz σ^2 bestimmt dabei die Distanz der umliegenden Pixel, die Einfluss auf den neuen Farbwert haben. Der neue Farbwert wird dabei durch die Mittelung der Farbwerte der umliegenden Pixel bestimmt, wobei je größer die Distanz, desto geringer ist der Einfluss auf den neuen Farbwert. Die Wahl einer großen Varianz σ^2 sorgt also dafür, dass das Bild sehr unscharf wird. Zur Untersuchung der Robustheit der Klassifikatoren gegenüber Gaussian Blur werden iterativ zwanzig Testsets erzeugt. Die Varianz σ^2 des Gaussian Blur ist dabei in dem Intervall $[0, 10]$ in 0.5er Schritten gestaffelt. [23][26]

4.2.4. Adversarial Attacks

In der KI werden unter Adversarial Attacks Verfahren verstanden, die das Ziel haben ML-Anwendungen zu korrumpieren. Es existieren verschiedene Angriffsverfahren, sowohl in der Trainingsphase als auch der Interferenzphase der Anwendungen. Eines der häufigsten

Angriffsverfahren stellen dabei Evasion Attacks dar. Formel 4.9 zeigt das mathematische Vorgehen bei der Durchführung solch eines Angriffes. Dabei wird eine valide Eingangsdatei x durch das Hinzufügen einer Pertubation n in ein Adversarial Example x' transformiert. [25][28][24]

$$x' = x + n \tag{4.9}$$

Die manipulierte Datei ist für den Menschen mit dem bloßen Auge kaum bis gar nicht vom Original zu unterscheiden, das KI Modell verleitet diese Pertubation jedoch zu einer fehlerhaften Prädiktion. Wie in Abbildung 4.5 zu sehen, wurde das ursprüngliche Objekt des MNIST Digits Datensatz korrekt von einem Klassifikator klassifiziert. Nachdem der Instanz eine bösartige Pertubation hinzugefügt wurde, kann diese jedoch nicht mehr korrekt klassifiziert werden. Die Intention des Angriffes kann dabei variieren. Hat der Angreifer das Ziel einer reinen Fehlklassifikation der Datei x', wird ungerichtet vorgegangen. Mathematisch kann das Ziel eines ungerichteten Angriffs anhand Formel 4.10 beschrieben werden. Formel 4.11 beschreibt das Vorgehen eines zielgerichteten Angriffs. Der Angreifer hat das Ziel, dass das Modell für das Adversarial Example x' ein spezifisches Klassenlabel y' vorhersagt. [28][24][25]

$$f(x') \neq y_{true} \tag{4.10}$$

$$f(x') = y' \tag{4.11}$$

Je nachdem, welche Informationen der Angreifer über sein Ziel besitzt, kann er unterschiedliche Angriffe durchführen. Evasion Attacks lassen sich in Abhängigkeit der verfügbaren Informationen in zwei Kategorien aufteilen:

- **White Box Attack**: Der Angreifer hat vollen Zugriff auf alle Informationen des Klassifikators. Dazu zählen unter anderem die Architektur und die Gewichtung des Modells des Klassifikatores, aber auch die getroffenen Vorhersagen. [25]
- **Black Box Attack**: In vielen Anwendungsfällen ist es für den Angreifer nicht umsetzbar, wichtige Informationen über das Modell zu erlangen. Der Angreifer muss deshalb nur anhand der Eingangsdaten und der zugehörigen Vorhersage eine geeignete Pertubation finden, die das Modell täuscht. Deshalb sind Blackbox Angriffe deutlich komplexer und zeitaufwendiger. [25]

Fast Gradiend Sign Method

In dieser Arbeit wurde die Fast Gradient Sign Method, kurz FGSM, angewandt, um adversielle Pertubationen zu erzeugen. Das FGSM ist ein White Box Attack. Wie in Formel (4.12) zu erkennen, wird der Instanz dabei eine adversielle Pertubation hinzugefügt. [24][28]

$$x' = x + \epsilon * sign(\nabla_x J(\theta, x, y)) \tag{4.12}$$

Zur Findung der adversiellen Pertubation wird dabei eine lineare Approximation der Loss-Funktion $J(\theta, x, y)$ durchgeführt. Die Loss-Funktion hat den Zweck, den Ausgangsfehler des Modells zu bewerten und setzt sich aus den Parametern des Modells , dem Input x und der Zielgröße y zusammen. Anhand dieser Funktion wird nach Pertubationen gesucht, die den Ausgangsfehler ansteigen lässt. Der Instanz x wird anschließend eine Pertubation der Größe ϵ in Richtung des Gradientenvektors hinzugefügt [49][39]. Der Gradient $\nabla_x J(, x, y)$, also die Änderung der Loss-Funktion in Abhängigkeit des Inputs x wird dabei mittels Backpropagation berechnet. Um die Funktionsweise des FGSM im Detail nachvollziehen zu können, wird es empfohlen, in folgender Fachliteratur nachzulesen: [24][28]

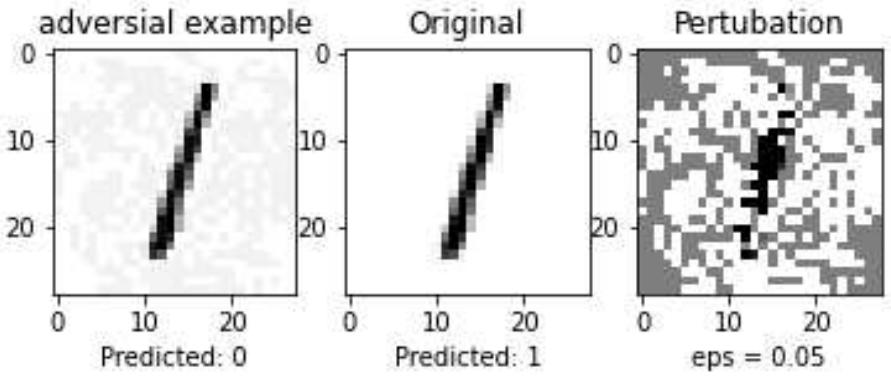


Abbildung 4.5.: Adversarial Example für den MNIST Digits, mittels FGSM erzeugt. (a) zeigt das Adversarial Example, (b) zeigt das Originalbild und (c) zeigt die durch FGSM erzeugte Pertubation.

Bei der Untersuchung der Adversarial Robustness wird von folgendem Szenario ausgegangen: Der Angreifer konnte nicht alle Informationen über eine Anwendung zur Klassifikation von Daten ergattern. Ihm gelang es lediglich zu erfahren, dass die Anwendung zur Klassifikation Ensemble-Verfahren einsetzt, die die Vorhersagen eines Set an Basisklassifikatoren kombiniert. Zudem konnte er Informationen über drei der Basisklassifikatoren sammeln. Er konnte erfahren, dass sowohl eine logistische Regression, eine Support Vector Machine als auch ein künstliches neuronales Netz als Basisklassifikatoren verwendet werden. Da für alle drei

Verfahren die FGSM zur Generierung Adversarial Examples genutzt werden kann, führt er diesen Angriff durch. Dabei probiert der Angreifer zuerst, ob es auch ausreicht, ein Adversarial Example anhand der Informationen von einem der drei Klassifikationsverfahren zu erzeugen. Anschließend versucht er es noch mit einem Angriff, der die Informationen aller drei Klassifikatoren enthält. Da der Angreifer mit diesem Angriff noch nicht vertraut ist, variiert er den Wert von ϵ für den MNIST Digits Datensatz in einem Intervall von $[0, 0.1]$ mit einer Schrittweite von 0.005. Für den MNIST Fashion Datensatz verwendet er ϵ Werte im Bereich $[0, 10]$.

5. Evaluierung der Klassifikatoren

In diesem Kapitel werden die Klassifikationsergebnisse der unterschiedlichen Datensätze anhand der zuvor vorgestellten Verfahren bewertet. Zunächst wird dabei untersucht, ob die verschiedenen Ensemble-Verfahren zu einer Verbesserung der Zuverlässigkeit und Robustheit führen können. Dabei wird bei den Blending-Verfahren insbesondere der Einfluss der Meta-Features untersucht. Anschließend wird noch bewertet, welches der vorgestellten Ensemble-Verfahren sich am besten dafür eignet, eine zuverlässige und robuste ML-Anwendung zu erzeugen. Da die relative Häufigkeit der Klassenlabels sowohl bei MNIST Digits als auch bei Fashion MNIST nahezu gleichverteilt ist, wird hier die Accuracy als entscheidende Metrik zur Bewertung der Zuverlässigkeit und Robustheit verwendet. Für die MedMNIST ist dies jedoch nicht gegeben. Es ist für einen sinnvollen Einsatz der Klassifikatoren essenziell, dass diese sowohl eine hohe Precision als auch einen guten Recall aufweisen können. Hier wird das F1-measure als entscheidende Metrik verwendet. Als Letztes wird anhand der zuvor vorgestellten Adversarial Attacks untersucht, ob Ensemble Methoden, die Adversarial Robustness einer Anwendung verbessern können.

5.1. Ergebnisse Voting

In Tabelle 4.1 sind die Klassifikationsergebnisse des Voting-Verfahrens und des besten Basisklassifikators für die sechs verschiedenen Datensätze dargestellt. Das Votingensemble konnte bei drei von sechs Klassifikationsaufgaben zu einem verbesserten Klassifikationsergebnis führen. Sowohl für den MNIST Digits $(+0,64\%)$ als auch für den MNIST Fashion $(+1,27\%)$ konnte die Accuracy verbessert werden. Bei den Datensätzen der MedMNIST Datenbank konnte lediglich für den MedMNIST Pneumonial Datensatz der F1-Score $(+1,85\%)$ verbessert werden. Für die anderen drei Datensätze hat sich die Klassifikationsleistung gegenüber dem besten Basisklassifikator verschlechtert. Je nach Datensatz beträgt die Reduktion zwischen -1,3 % und -4,42 %. Auffällig ist, dass für jede Klassifikationsaufgabe die Precision verbessert werden konnte. Die Spanne der Verbesserung beträgt dabei je nach Datensatz zwischen 0.65 % und 7.24 %. Der Recall konnte lediglich für den MNIST Digits Datensatz um 0,66 % erhöht werden. Für die restlichen Datensätze verschlechterte sich um bis zu -13.10 %.

Tabelle 5.1.: Vergleich der Klassifikationsergebnisse des Votingschemas mit dem besten Basisklassifikator.

Datensatz	Algorithmus	accuracy	precision	recall	f1-measure
MNIST Digits	Logistic Regression	92.39	92.28	92.30	92.28
	Voting	**93.03**	**92.94**	**92.96**	**92.93**
MNIST Fashion	Logistic Regression	84.05	84.05	83.94	83.97
	Voting	**85.32**	**85.32**	**85.21**	**85.22**
MedMNIST OrganA	Convolutional Neural Networks	**63.35**	61.04	**63.74**	**62.36**
	Voting	50.75	**67.72**	50.64	57.95
MedMNIST OrganC	Random Forest	**66.51**	63.18	**71.77**	**67.20**
	Voting	64.08	**70.17**	62.11	65.90
MedMNIST Breast	Stochastic Gradient Descent	78.21	70.05	**72.22**	70.94
	Voting	**80.13**	**75.20**	71.37	**72.79**
MedMNIST Pneumonial	Convolutional Neural Networks	**84.78**	81.58	**85.71**	**82.89**
	Voting	84.29	**88.82**	79.40	81.40

Es lässt sich festhalten, dass mittels Voting die Klassifikationsleistung verbessert werden kann. Der Erfolg hängt stark von der Zusammensetzung der Basisklassifikatoren ab. Es wird ein hoher Grad an Diversität benötigt (eine paarweise Diversität zwischen den Klassifikatoren ist dabei nicht ausreichend). Dabei ist die Gesamtdiversität des Ensembles entscheidend. Die Fehler der Klassifikatoren müssen dabei weitestgehend unkorreliert sein. Das Klassifikationsergebnis kann nur verbessert werden, wenn die Mehrheit der Klassifikatoren nicht dieselben Objekte fehlerhaft klassifizieren. Da die Basisklassifikatoren anhand der gleichen Trainingsdaten trainiert wurden, existiert eine Korrelation im Hypothesenraum der Klassifikatoren. Ist die Mehrheit der Klassifikatoren nicht in der Lage, bestimmte Objekte bzw. Klassen sicher zu klassifizieren, scheitert das Kollektiv. Dadurch können Objekte, die von einem einzelnen Klassifikator richtig zugeordnet worden wären, vom Ensemble falsch klassifiziert werden.

Ein Ansatz, dieses Problem abzuschwächen, ist der Einsatz von labelspezifischen Gewichtungen der Klassifikatoren. So kann im Entscheidungsprozess berücksichtigt werden, dass ein Klassifikator nicht die Fähigkeit besitzt, ein bestimmtes Label zu erkennen. Des Weiteren sollten im Trainingsprozess der Klassifikatoren Maßnahmen getroffen werden, um die Gesamtdiversität des Ensembles zu erhöhen. Unter anderem kann die Diversität der Klassifikatoren durch Cross Validation oder Bootstrap Aggregation erhöht werden.

Bei der Betrachtung der Klassifikationsergebnisse für die erzeugten Fault Injections (Abbildung

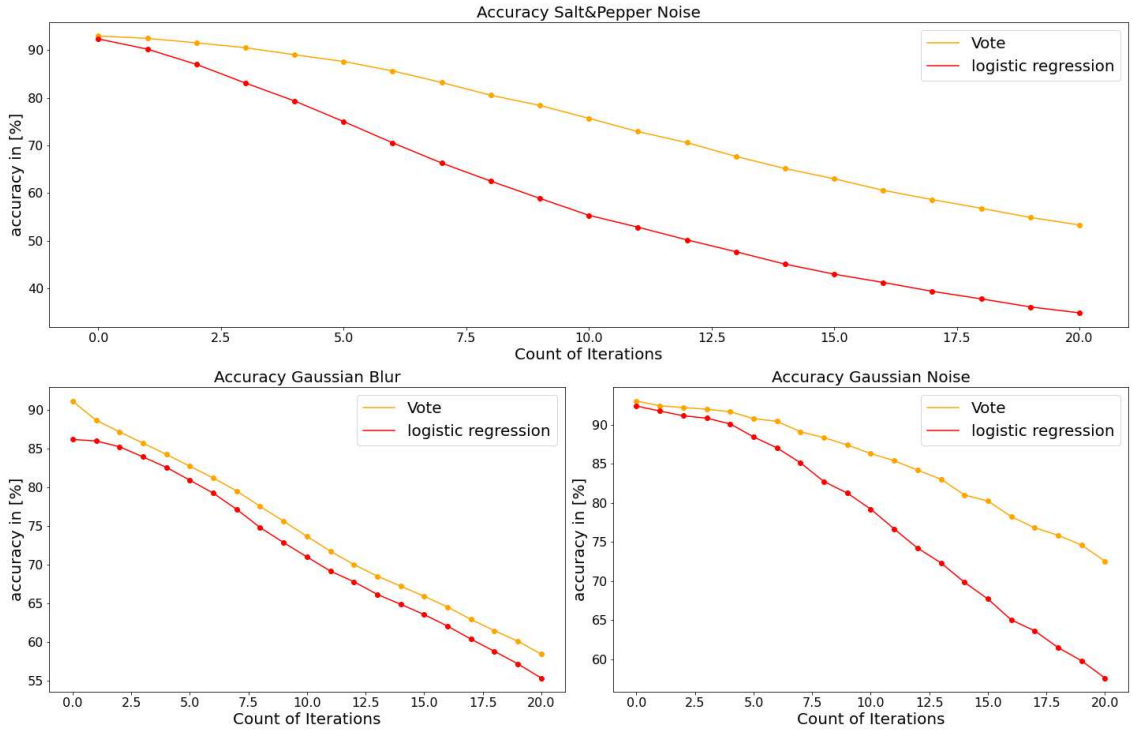


Abbildung 5.1.: Robustheit des Voting Verfahren gegenüber Salt & Pepper Noise (a), Gaussian Blur (b) und Gaussian Noise(c). Dabei wurde iterativ die Stärke der Störungen erhöht. Als Datensatz wurden die Testdaten des MNIST Digits verwendet.

5.1) lässt sich erkennen, dass ein Votingschema ein geeignetes Verfahren zur Steigerung der Robustheit darstellen kann. Gerade bei schwachen Störungen in den Inputdaten überzeugt die Leistung des Kollektivs. Im ML kann ein Voting Ensemble dabei als redundantes System betrachtet werden. Unter der Voraussetzung, dass ohne Störung das Objekt richtig klassifiziert worden wäre, kann ein Voting-Verfahren solange ein valides Klassifikationsergebnis liefern, solange die Mehrheit der Basisklassifikatoren von der auftretenden Störung nicht beeinträchtigt ist. Essenziell für den Erfolg des Kollektivs ist demnach die Sensitivität der Klassifikatoren auf eine Störung. Beeinträchtigt eine Störung die Mehrheit der Basisklassifikatoren, kann die Robustheit des Systems nicht mehr gewährleistet werden.

Ein interessanter Ansatz, um die Robustheit des Ensembles zu maximieren, ist das gezielte Adversarial Training einzelner Klassifikatoren. Es wird sichergestellt, dass die Mehrheit der Klassifikatoren unsensibel gegenüber einer Störung sind, ohne die Gesamtzuverlässigkeit eines Systems zu stark zu beeinträchtigen.

5.2. Ergebnisse Blending mittels logistischer Regression

Der Tabelle 4.3 können die Klassifikationsergebnisse der Blending Verfahren mittels logistischer Regression entnommen werden. MLR konnte für alle sechs Klassifikationsaufgaben das Klassifikationsergebnis gegenüber dem besten Basislerner verbessern. Für den MNIST Digits Datensatz konnte eine Accuracy von $94,71\%$ $(+2,32\%)$ erreicht werden und bei den Vorhersagen für den MNIST Fashion konnte eine Accuracy von $86,57\%$ erzielt werden. Gegenüber der logistischen Regression verbesserte sich das Klassifikationsergebnis damit um 2.52%. Auch für die MedMNIST Datensätze ist eine signifikante Steigerung des F1 Score zu erkennen. Dabei verbesserte sich der F1-Score je nach Datensatz um $0.09\% - 6.42\%$, wobei die größte Verbesserung für den MedMNIST OrganA Datensatz $(+6.42\%)$ erreicht wurde.

Die Klassifikationsergebnisse von MLR2 und MLR3 konnten hingegen nicht überzeugen. Lediglich für die beiden binären Klassifikationsaufgaben konnten zufriedenstellende Klassifikationsergebnisse erzielt werden. Für den MedMNIST Breast Datensatz konnte MLR2 den F1-Score um 1.55% und MLR3 sogar um 2.16% verbessern. Für diesen Datensatz konnten sowohl MLRünd ML3 ein besseres Klassifikationsergebnis erzielen als MLR(+1.09 %). Auch für den MedMNIST Pneumonial Datensatz führte die Kombination mittels MLR3 zu einer besseren Performance (0.84%). Bei MLR2 verschlechterte sich der F1-Score jedoch um (-0.12%). Für die restlichen vier Datensätze konnte jedoch keines der beiden Verfahren das Klassifikationsergebnis verbessern. Im Gegenteil, der Einsatz der beiden Verfahren verschlechterte sogar die Klassifikationsergebnisse. Für MNIST Digits und Fashion verschlechterte sich die Accuracy zwischen -8.67% und -11.78%. Der F1-Score rutschte sogar um bis zu -22.06% ab.

Das schlechte Abschneiden des Blendings mittels MLR2 lässt sich damit begründen, dass die Meta-Features weniger Informationen enthalten als bei MLR und MLR3. Bei der Kombination der Vorhersagen wird die Konfidenz der Basisklassifikatoren nicht berücksichtigt. Bei MLR3 wird die Konfidenz anhand der Entropie der Vorhersagen berücksichtigt. Die Entropie hat jedoch den Nachteil, dass es nicht ersichtlich ist zwischen welchen Klassenlabels ein Klassifikator „schwankt “.

Nur die Basisklassifikatoren von MLR geben anhand der Class Probabilities Auskunft über die A-posteriori Wahrscheinlichkeiten der Labels. Damit steht dem Meta-Learner nicht nur die Information über die Konfidenz der Hypothesen zur Verfügung, sondern auch, welche Klassenlabels als Vorhersage in Betracht gezogen werden. Ein weiterer Nachteil von MLR2 und MLR3 beruht in der Struktur der logistischen Regression. Beide Verfahren nutzen diskrete Attribute als Meta-Features. Eine Änderung in der Vorhersage eines Basisklassifikators hat eine große Auswirkung auf die systematischen Komponenten und somit auch eine starke Auswirkung auf die A-posteriori Wahrscheinlichkeit. Der Wert der systematischen Komponente weist statt einer kontinuierlichen Kurve ein sprunghaftes Verhalten auf. Das Verhalten der systematischen Komponenten von MLR ist hingegen deutlich dynamischer. Da die Meta-Features

kontinuierliche Werte im Bereich $[0, 1]$ annehmen, weist auch die Diskriminanzfunktion ein kontinuierliches Verhalten auf.

Ein weiterer Nachteil von MLR2 und MLR3 ist auch auf die Anzahl an Meta-Features zurückzuführen. MLR2 verwendet 7 und MLR3 14 Attribute. Wird im Trainingsprozess eine fehlerhafte Gewichtung erlernt, hat dies große Auswirkungen auf den Bias. Verwendet das Modell hingegen eine größere Anzahl an Meta-Featuren, hat eine falsche Gewichtung einzelner Features eine geringere Auswirkung auf die Qualität des Klassifikators. Anhand dieser Annahme kann begründet werden, weshalb MLR für binäre Klassifikationsaufgaben MLR2 und MLR3 nicht outperformen kann. Für binäre Klassifikationsaufgaben verwendet MLR nämlich auch nur 14 Meta-Features.

Für einen effizienten Einsatz der logistischen Regression als Meta-Learner sollten bei der Wahl der Meta-Features möglichst Features mit kontinuierlichen Werten genutzt werden, die bestenfalls Informationen über die Konfidenz der Basisklassifikatoren enthalten. Zudem sollte eine ausreichende Anzahl an Meta-Features genutzt werden. Von den vorgestellten drei Verfahren empfiehlt sich deshalb nur MLR als ein geeignetes Blending-Verfahren.

Auch die Untersuchung der Robustheit der Blending-Verfahren gegenüber den implementierten Fault Injections bestätigen diese Annahmen. Abbildung 5.2 zeigt die Accuracy der drei Blending-Verfahren MLR,MLR2 und MLR3 für die Fault Injections im Vergleich zur Accuracy der logistischen Regression. Dabei ist auffällig, dass sowohl MLR2 und MLR3 sehr empfindlich auf Pertubationen in den Daten reagieren. Es zeigt sich, dass schon kleine Störungen enorme Auswirkungen auf das Klassifikationsergebnis haben. Es kann keine Verbesserung der Robustheit festgestellt werden. Im Gegenteil dazu weist die logistische Regression trotz Störungen eine bessere Performance als die beiden anderen Verfahren auf. Der Einsatz von MLR zur Klassifikation bezweckt hingegen eine deutliche Verbesserung der Robustheit eines Klassifikationsverfahrens.

So kann MLR trotzt Salt & Pepper Noise oder Gaussian Noise eine bessere Klassifikationsleistung erzielen. Beeinflusst die auftretende Störung einen Klassifikator negativ, hat dies Auswirkung auf seine Class Probabilities. Die Konfidenz der Vorhersage sinkt, was in der Entscheidungsfindung berücksichtigt wird. Weist die Vorhersage eines Klassifikators eine geringe Konfidenz auf, wird diese bei der Entscheidungsfindung geringer berücksichtigt. Der Ausfall einzelner Klassifikatoren aufgrund einer Störung kann also vom Modell kompensiert werden. Führt eine Störung bei einer großen Anzahl der Basisklassifikatoren zu einer Fehlklassifikation, kann die Instanz hingegen nicht mehr korrekt klassifiziert werden.

Abschießend kann für das Meta-Learning mittel logistischer Regression zusammengefasst werden, dass von den drei vorgestellten Zusammensetzungen der Inputfeatures die Class Probabilities der Basisklassifikatoren bevorzugt genutzt werden sollte, da sowohl eine bessere

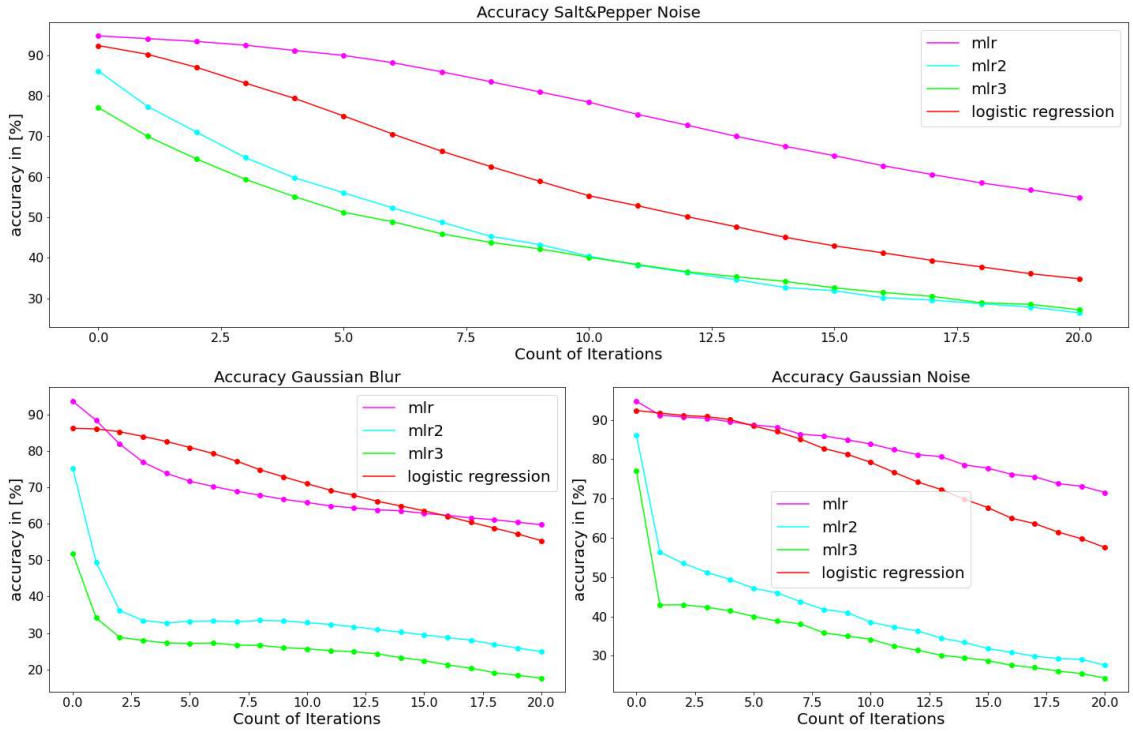


Abbildung 5.2.: Robustheit der Blending Verfahren mit logistischer Regression als Meta-Learner für Salt & Pepper Noise(a), Gaussian Blur(b) und Gaussian Noise(c). Dabei wurde iterativ die Stärke der Störungen erhöht. Als Datensatz wurden die Testdaten des MNIST Digits verwendet.

Zuverlässigkeit als auch eine bessere Robustheit erreicht werden konnte. Sollte dieses Verfahren jedoch keine akzeptable Güte liefern, ist es ratsam, für binäre Klassifikationsaufgaben auch die anderen beiden Meta-Feature Kombinationen zu implementieren.

Tabelle 5.2.: Accuracy, Precision, Recall & f1-measure der Blending Verfahren mit Logistscher Regression als Meta-Learner im Vergleich zum besten Basisklassifikators.

Datensatz	Algorithmus	accuracy	precision	recall	f1-measure
MNIST Digits	Logistic Regression	92.39	92.28	92.30	92.28
	Meta Logistic Regression	**94.71**	**94.64**	**94.66**	**94.65**
	Meta Logistic Regression2	86.30	86.09	86.16	86.06
	Meta Logistic Regression3	77.35	76.92	76.99	76.88
MNIST Fashion	Logistic Regression	84.05	84.05	83.94	83.97
	Meta Logistic Regression	**86.57**	**86.57**	**86.48**	**86.50**
	Meta Logistic Regression2	75.38	75.38	74.56	74.37
	Meta Logistic Regression3	73.35	73.35	73.41	73.32
MedMNIST OrganA	Convolutional Neural Networks	63.35	61.04	63.74	62.36
	Meta Logistic Regression	**67.07**	**66.28**	**71.49**	**68.79**
	Meta Logistic Regression2	51.57	47.83	52.24	49.94
	Meta Logistic Regression3	54.26	54.62	60.68	57.49
MedMNIST OrganC	Random Forest	66.51	63.18	71.77	67.20
	Meta Logistic Regression	**71.14**	**69.49**	**74.30**	**71.81**
	Meta Logistic Regression2	48.73	45.67	44.62	45.14
	Meta Logistic Regression3	55.42	53.58	59.30	56.29
MedMNIST Breast	Stochastic Gradient Descent	78.21	70.05	72.22	70.94
	Meta Logistic Regression	78.85	71.24	73.10	72.03
	Meta Logistic Regression2	78.85	71.99	73.08	72.49
	Meta Logistic Regression3	**79.49**	**72.43**	**73.95**	**73.10**
MedMNIST Pneumonial	Convolutional Neural Networks	84.78	81.58	85.71	82.89
	Meta Logistic Regression	85.26	81.20	**88.04**	82.98
	Meta Logistic Regression2	85.10	80.98	87.93	82.78
	Meta Logistic Regression3	**85.74**	**82.09**	87.81	**83.73**

5.3. Ergebnisse Blending mittels Decision Tree

In Tabelle 4.2 sind die Klassifikationsergebnisse der Blending-Verfahren mit Decision Tree als Meta-Learner dargestellt. Im direkten Vergleich der drei Meta-Feature Möglichkeiten fällt auf, dasss MDT zwar nicht für alle Klassifikationsaufgaben die besten Klassifikationsergebnisse erzielen konnte, jedoch die konstanteste Leistung. MDT verbessert gegenüber dem besten Basisklassifikator die Accuracy für den MNIST Digits Datensatz um $0,39\%$ und Fashion MNIST um $1,10\%$. Auch für alle Datensätze der MedMNIST Datenbank konnte der F1-Score durch MDT verbessert werden. Je nach Datensatz verbesserte sich dieser um $0,62\% - 2,09\%$. Mithilfe des MDT2 konnte ebenfalls die Accuracy für MNIST Digits (+0,32 %) und Fashion MNIST (+0,93 %) verbessert werden. Anhand der MedMNIST Datensätze zeigt sich, dass der Erfolg von MDT2 stark von der Klassifikationsaufgabe und den Basisklassifikatoren abhängt. Für MedMNIST OrganA und OrganC konnte der F1-Score verbessert werden. Für die beiden binären Klassifikationsaufgaben verschlechterte sich das Klassifikationsergebnis. Der erzielte F1-Score reduzierte sich um $-2,93\%$ (Pneumonial) und $-4,56\%$ (Breast). Anwendungen, die MDT3 verwenden, können auch keine Verbesserung der Klassifikationsleistung garantieren. Ob eine Verbesserung erzielt werden kann, ist dabei stark von der Klassifikationsaufgabe abhängig. So verbesserte sich die Accuracy für den MNIST Digits Datensatz um $0,54\%$ und für den Fashion MNIST Datensatz verschlechterte sie sich (-0,90 %). Der F1-Score konnte für drei von vier MedMNIST Datensätzen verbessert werden. Je nach Datensatz verbesserte sich der F1-Score um $0.01\% - 2,07\%$. Für den MedMNIST Breast konnte aber kein sinnvolles Kombinationsschema entwickelt werden, der F1-Score verschlechterte sich um $-7,49\%$.

Alle der drei vorgestellten Verfahren sind in der Lage, das Klassifikationsergebnis zu verbessern. Anders als die Logistische Regression kann ein Decision Tree sowohl kontinuierliche als auch diskrete Features gut verarbeiten. MDT konnte die konstantesten Ergebnisse bei der Verbesserung der Zuverlässigkeit vorweisen. Bei der Analyse der Robustheit der Blending-Verfahren mittels Decision Tree zeigt sich ein entscheidender Vorteil des MDT3 gegenüber den anderen beiden Verfahren. In Abbildung 5.3 sind die Ergebnisse der Untersuchung der Robustheit der drei Blending-Verfahren für den MNIST Digits Datensatz dargestellt. Es ist eindeutig zu erkennen, dass MDT3 für die untersuchten Fault Injections die besten Klassifikationsergebnisse erzielen kann. Für alle der drei Störungen konnte die Accuracy durch MDT3 verbessert werden. Mit den anderen beiden Verfahren (MDT & MDT2) ist es nur bedingt möglich, die Robustheit zu steigern. Für Salt & Pepper Noise konnte die Robustheit ebenfalls verbessert werden. Für Daten, die durch ein Gaussian Noise oder Gaussian Blur gestört sind, versagen die Verfahren. Die Verbesserung der Robustheit durch MDT und MDT3 ist darauf zurückzuführen, dass die Meta-Features Informationen über die Konfidenz der Basisklassifikatoren enthalten. Es zeigt sich, dass die Vorhersage eines Klassifikators, der durch eine Störgröße negativ beeinflusst wird, häufig eine geringere Konfidenz besitzt. Da MDT2 die Konfidenz nicht in den Meta-Features berücksichtigt, steht ihm diese Information nicht

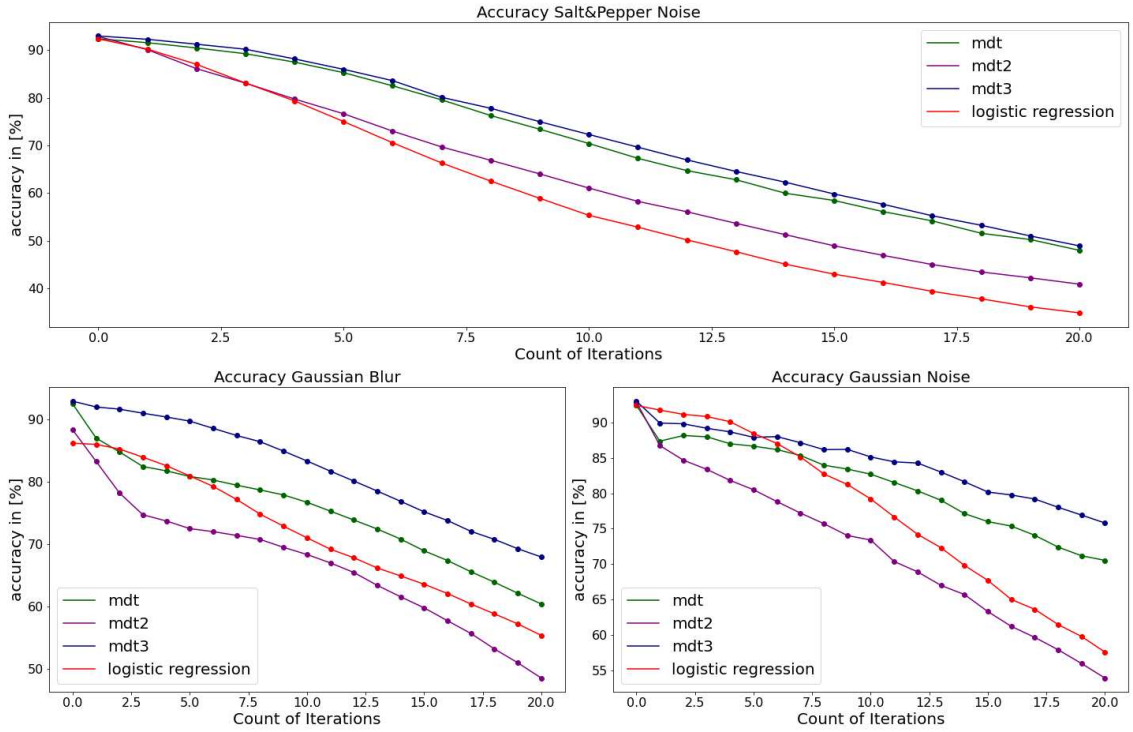


Abbildung 5.3.: Robustheit der Blending Verfahren mit Decision Tree als Meta-Learner für Salt &Pepper Noise(a), Gaussian Blur(a) und Gaussian Noise(a). Dabei wurde iterativ die Stärke der Störungen erhöht. Als Datensatz wurden die Testdaten des MNIST Digits verwendet.

zur Verfügung. Der Vorteil von MDT3 gegenüber MDT lässt sich auf die Eigenschaften des verwendeten Decision Tree (CART) Algorithmus zurückführen. Dieser nutzt in jedem Knoten nur ein einzelnes Feature, um eine Instanz einer Teilmenge zuzuordnen. Die Konfidenz eines Basisklassifikators von MDT ergibt sich aus der Summe der Class Probabilities. Somit kann in einem Knoten die Konfidenz eines Basisklassifikators nicht allgemein abgefragt werden, sondern nur die A-posteriori Wahrscheinlichkeit für ein Klassenlabel. Bei MDT3 ist die Konfidenz der Vorhersage eines Basisklassifikators in einem einzelnen Meta-Feature gespeichert. Es kann also anhand eines einzelnen Knoten ermittelt werden, ob ein Basisklassifikator durch eine Störung betroffen ist.

Abschließend lässt sich sagen, dass alle drei Meta-Feature Selektionen die Zuverlässigkeit einer Anwendung verbessern können. Für eine Klassifikationsaufgabe sollten zunächst alle drei Verfahren implementiert und untersucht werden, welches sich am besten zur Verbesserung der Zuverlässigkeit einsetzten lässt. Haben alle drei Verfahren einen positiven Einfluss auf das Klassifikationsergebnis, sollte auf MDT3 zurückgegriffen werden, da nur mit diesem Verfahren die Robustheit nachhaltig verbessert werden kann. Deshalb wird für den weiteren Vergleich

der Ensemble-Verfahren MDT3 genutzt.

Tabelle 5.3.: Accuracy, Precision, Recall & f1-measure der Blending Verfahren mit Decision Tree als Meta-Learner im Vergleich zum besten Basisklassifikators.

Datensatz	Algorithmus	accuracy	precision	recall	f1-measure
MNIST Digits	Logistic Regression	92.39	92.28	92.30	92.28
	Meta Decision tree	92.78	92.68	92.72	92.69
	Meta Decision tree 2	92.71	92.61	92.62	92.60
	Meta Decision tree 3	**92.93**	**92.84**	**92.87**	**92.85**
MNIST Fashion	Logistic Regression	84.05	84.05	83.94	83.97
	Meta Decision tree	**85.15**	**85.15**	**85.36**	**85.20**
	Meta Decision tree 2	84.98	84.98	84.95	84.93
	Meta Decision tree 3	83.15	83.15	82.98	83.03
MedMNIST Organ A	Convolutional Neural Networks	63.35	61.04	63.74	62.36
	Meta Decision tree	61.33	60.59	**65.58**	62.99
	Meta Decision tree 2	**63.68**	**62.29**	65.33	**63.77**
	Meta Decision tree 3	60.86	60.81	64.01	62.37
MedMNIST OrganC	Random Forest	66.51	63.18	71.77	67.20
	Meta Decision tree	67.90	65.76	**72.24**	68.84
	Meta Decision tree 2	**68.69**	**67.92**	71.04	**69.44**
	Meta Decision tree 3	67.78	65.70	69.80	67.69
MedMNIST Breast	Stochastic Gradient Descent	**78.21**	70.05	**72.22**	70.94
	Meta Decision tree	75.64	**74.31**	70.79	**71.73**
	Meta Decision tree 2	75.00	65.60	67.64	66.38
	Meta Decision tree 3	65.38	68.80	64.80	63.20
MedMNIST Pneumonial	Convolutional Neural Networks	84.78	81.58	85.71	82.89
	Meta Decision tree	**86.70**	**83.46**	**88.34**	**84.98**
	Meta Decision tree 2	82.85	78.25	85.77	79.96
	Meta Decision tree 3	86.54	83.68	87.49	84.96

5.4. Vergleich der Ensembleverfahren untereinander

Die Tabelle 5.3 zeigt die durch die Ensemble-Verfahren erreichte Verbesserung in der Accuracy, Precision, Recall und F1- Measure. Es zeigt sich, dass sowohl Blending- als auch Voting-Verfahren in der Lage sind, die Zuverlässigkeit zu verbessern. Dabei konnten Blending das Voting outperformen. Es konnte häufiger eine Verbesserung des Klassifikationsergebnisses festgestellt und eine bessere prozentuale Verbesserung erreicht werden.

Mit MLR konnte für alle sechs Klassifikationsaufgaben Kombinationschemata entwickelt werden, die die Klassifikationsleistung verbessern. Das Voting-Verfahren konnte hingegen nur drei von 6 ML-Anwendungen verbessern. Neben der konstantesten Performance kann MLR auch die größten Verbesserungen bewirken. Je nach Datensatz verbesserte sich die Accuracy um $0.48\% - 4.63\%$ und der F1-Score um $0.09\% - 6.42\%$. Die Performance von MDT3 und Voting ist deutlich schwächer. MDT3 konnte die Accuracy maximal um 1.76% verbessern. Es gibt jedoch auf Fälle, bei denen kein sinnvolles Kombinationsschema erlernt werden konnte. Die Accuracy verschlechterte sich um bis zu -12.82% (Breast). Der F1-Score weißt ein ähnliches Verhalten auf. Je nach Datensatz veränderte sich das Klassifikationsergebnis um -7.42%bis $+2.07\%$. Dabei ist anzumerken, dass auch die anderen Blending-Verfahren mit Decision Tree als Meta-Learner ähnliche Ergebnisse erzielen. Das Votingschema verbesserte die Accuracy um bis zu $1,27\%$. Auch bei Voting-Verfahren kann es vorkommen, dass sich die Accuracy verschlechtert. Für den Organ A konnte ein Verlust an Accuracy um -12.61% festgestellt werden. Der F1-Score änderte sich durch Voting um $-4.42\% - +1.85\%$.

Die Gewichtung der Stimmen der Basisklassifikatoren anhand von Accuracy oder F1-Score eignet sich somit nur bedingt zur Verbesserung der Klassifikationsleistung durch Voting. Blending eignet sich hingegen deutlich besser, um die Performance der Anwendung zu verbessern. Es kann auch ein Klassenlabel korrekt vorhergesagt werden, wenn mehr als die Hälfte der Basisklassifikatoren eine fehlerhafte Prädiktion abgibt. Der Erfolg eines Kombinationsverfahrens ist dabei jedoch stark abhängig von der Diversität der Basisklassifikatoren, Wahl der Metafeatures und Wahl des Meta-Learners.

In Abbildung 5.4 sind die Ergebnisse der Untersuchung der Robustheit für den MNIST Digits Datensatz dargestellt. Die drei Graphen zeigen die erreichte Accuracy der Ensemble-Verfahren (MLR, MDT3 und Voting) für die unterschiedlichen Fault Injections. Als Referenzwert wird die erreichte Accuracy des „besten“ Basisklassifikators verwendet, in diesem Fall die der logistischen Regression. Auf den ersten Blick zeigt sich, dass alle drei Ensemble-Verfahren die Robustheit der Anwendung steigern. Dabei konnten Voting und MLR3 für alle drei Fault Injection die Accuracy verbessern. MLR hingegen nur für das Salt & Pepper Noise und das Gaussian Noise. Für das Gaussian Blur konnte keine Verbesserung festgestellt werden. Beim direkten Vergleich zeigt sich, dass die Performance der Ensemble-Verfahren stark von der auftretenden Störung und ihrer Stärke abhängt. Logischerweise kommt es zu mehr Fehlklassifikationen, je

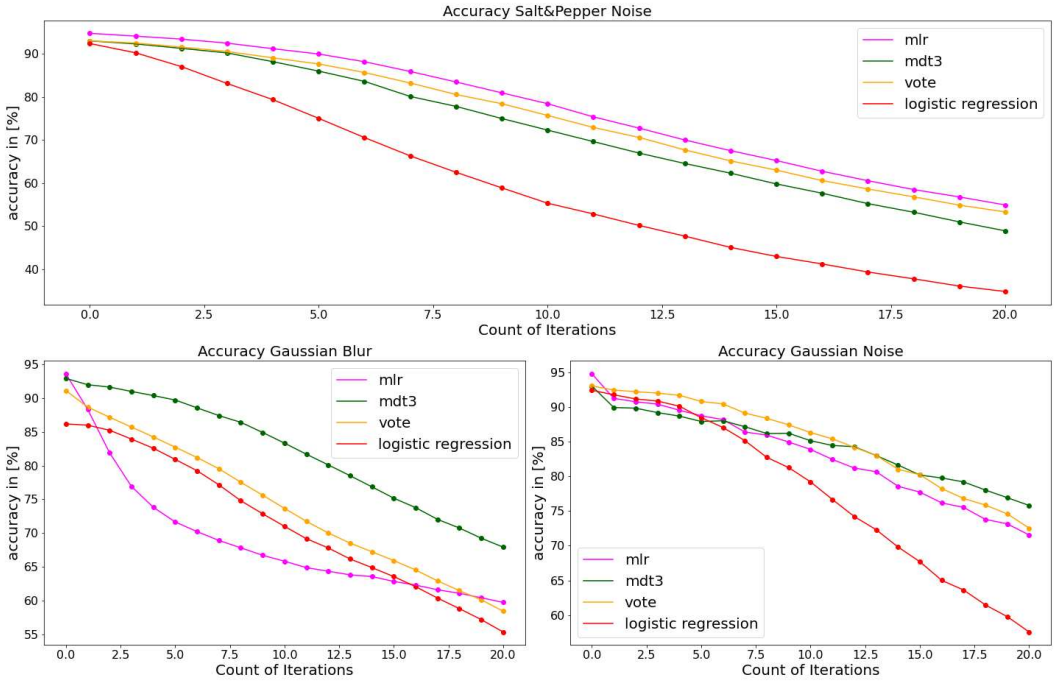


Abbildung 5.4.: Vergleich der Robustheit der Ensemble-Verfahren gegenüber Salt & Pepper Noise(a), Gaussian Blur(a) und Gaussian Noise(a). Dabei wurde iterativ die Stärke der Störungen erhöht. Als Datensatz wurden die Testdaten des MNIST Digits verwendet.

stärker die Störung ist. Es zeigt sich das die Kombinationsverfahren unterschiedlich anfällig für bestimmte Störungen sind. MLR und Voting sind dabei nicht so anfällig gegenüber Salt & Pepper Noise wie MDT3. Jedoch sind die Klassifikationsergebnisse für Gaussian Blur von MDT3 signifikant besser als von MLR und Voting. Bei Gaussian Noise in den Inputdaten können Voting und MDT3 besser Klassifikationsergebnisse aufweisen als MLR. Allgemein zeigt sich das MDT3 in der Lage ist die robusteste Anwendung zu erzeugen. Im Anhang in Abbildung B.4 sind die Ergebnisse der Untersuchung der Robustheit für den Fashion MNIST Datensatz dargestellt. Hier lässt sich ein ähnliches Verhalten der Ensemble-Verfahren gegenüber den Fault Injections erkennen, dies bestätigt die getroffenen Annahmen.

Es lässt sich abschließend sagen, dass Blending Verfahren ein besseres Verfahren zur Kombination von Basisklassifikatoren sind, als Voting verfahren. Mittels Blending können sowohl zuverlässigere als auch robustere Anwendungen erzeugt werden als durch Voting. Die Qualität der Blending-Verfahren hängt dabei jedoch stark von der Wahl der Meta-Features und des Meta-Learners ab. Es hat sich gezeigt, dass zur Verbesserung der Zuverlässigkeit logistische Regression als Meta-Learner und Class Probabilities als Meta-Features gewählt werden sollten. Hat die Robustheit hingegen Priorität, sollte Decision Tree als Meta-Learner mit den Entropien

und den Klassen Labels mit der höchsten A-posteriori Wahrscheinlichkeit als Meta-Features gewählt werden. Es ist zudem anzumerken, dass keine allgemeingültige Garantie über die Verbesserung der Robustheit gegeben werden kann. Sind die Basisklassifikatoren zu sensible gegenüber der Störung, kann auch ein Blending Verfahren nicht vor Fehlklassifikationen schützen.

Tabelle 5.4.: Accuracy, Precision, Recall & f1-measure der Esemble Verfahren (Voting, MLR, MDT3) für die unterschiedlichen Datensätze.

Datensatz	Algorithmus	accuracy	precision	recall	f1-measure
MNIST Digits	Voting	0.64	0.65	0.66	0.65
	MDT3	0.54	0.55	0.58	0.57
	MLR	**2.32**	**2.36**	**2.36**	**2.37**
MNIST Fashion	Voting	1,27	1,27	1,27	1,24
	MDT3	-0.90	-0.90	-0.96	-0.95
	MLR	**2,52**	**2,52**	**2,54**	**2,52**
MedMNIST OrganA	Voting	-12.61	**6.68**	-13.10	-4.42
	MDT3	-2.50	-0.23	0.27	0.01
	MLR	**3.71**	5.24	**7.75**	**6.42**
MedMNIST OrganC	Voting	-2.43	**6.99**	-9.66	-1.30
	MDT3	1.27	2.52	-1.97	0.49
	MLR	**4.63**	6.31	**2.54**	**4.62**
MedMNIST Breast	Voting	-0.48	**5.15**	-0.86	**1.85**
	MDT3	**-12.82**	-1.25	-7.42	-7.74
	MLR	0.64	1.19	**0.88**	1.09
MedMNIST Pneumonial	Voting	-0.48	**7.24**	-6.31	-1.49
	MDT3	**1.76**	2.09	1.78	**2.07**
	MLR	0.48	-0.38	**2.33**	0.09

5.5. Adversarial Robustness

Abbildung 5.6 zeigt die Ergebnisse der Untersuchung für den MNIST Digits Datensatz. Als Angriff wurde der FGSM eingesetzt. Graph (a) zeigt die Adversarial Robustness gegenüber Adversarial Examples, die anhand Informationen über das Modell der logistischen Regression erzeugt wurden. Für Graph (b) wurden die Informationen über das Neuronale Netz genutzt und für (c) die Informationen der Support Vector Machine. Für alle Angriffe wurde dabei der Epsilon-Wert iterativ erhöht. Es werden Epsilon-Werte im Bereich $[0, 0.1]$ verwendet. Es zeigt sich, dass alle Ensemble-Verfahren für alle durchgeführten Angriffe die Adversarial Robustness verbessern können. Gerade bei dem Adversarial Attack auf die Support Vector Machine lässt sich ein signifikanter Unterschied in den Klassifikationsergebnissen erkennen. Nur für große ϵ Werte ist das neuronale Netz unempfindlicher auf den Angriff als die Ensemble-Verfahren. Allgemein kann besonders für kleine ϵ Werte,also extrem schwachen Pertubationen, die Zuverlässigkeit durch den Einsatz von Ensemble Methoden gesichert werden. Wie bei den anderen Fault Injections weist das Blending-Verfahren MDT3 die größte Robustheit auf. Bei der Untersuchung der Adversarial Robustness für den Fashion MNIST Datensatz ist ein ähnliches Verhalten der Basisklassifikatoren festzustellen. Es ist zu beachten, dass größere ϵ Werte benötigt werden, um Fehlklassifikationen zu bezwecken.

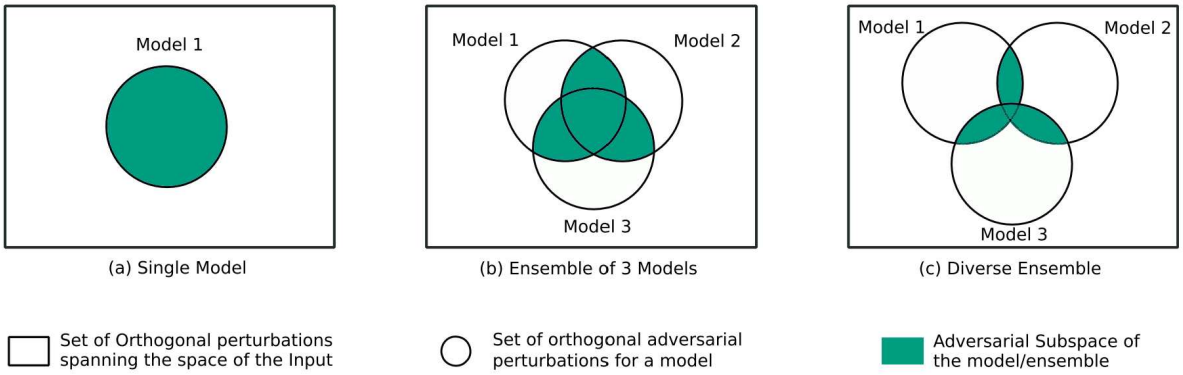


Abbildung 5.5.: Adversarial Robustness durch Diversität (In Anlehnung an [28]). (links) zeigt die Menge an adversiellen Pertubationen für einen einzelnen Klassifikator, (mitte) für ein Ensemble aus drei Klassifikatoren und (rechts) für ein diverses Ensemble.

Nach dem Paper von Sanjay Kariyappa und Moinuddin K.Qureshi lässt sich die Adversarial Robustness eines Ensembles auf die Diversität der Basisklassifikatoren zurückführen [28]. Es wird angenommen, dass für einen einzelnen Klassifikator eine endliche Menge an möglichen Pertubationen zur Täuschung eines Klassifikators existieren. Dieser Raum wird auch Adversarial Subspace (Adv-SS) genannt. Wird ein einzelner Klassifikator verwendet, reicht es, wenn

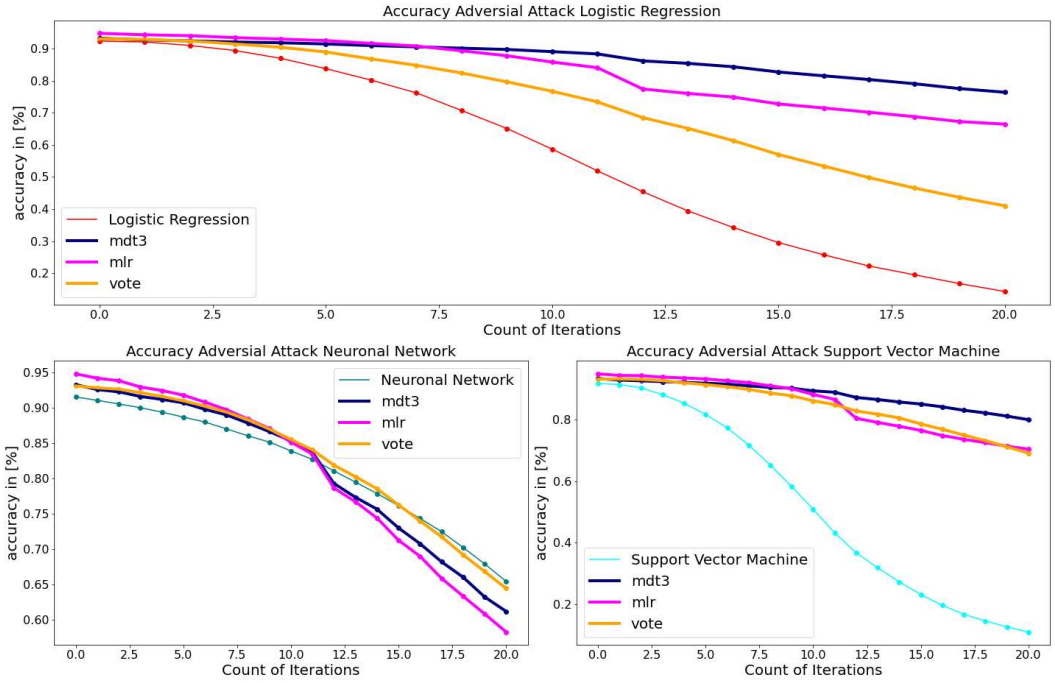


Abbildung 5.6.: Robustheit der Ensemble-Verfahren gegenüber Adversarial Example die mittels FGSM erzeugt. Die Adversarial Examples wurden anhand der Informationen über das Modell der logistischen Regression (a), Neuronale Netz (b) und der Support Vector Machine (c) erzeugt. Dabei wurde iterativ ϵ erhöht. Als Datensatz wurden die Testdaten des MNIST Digits verwendet.

der Angreifer es schafft, eine der möglichen Pertubationen zu generieren. Wird ein Ensemble aus unterschiedlichen Klassifikatoren verwendet, muss eine Pertubation erzeugt werden, die die Mehrheit der Klassifikatoren täuscht. Es muss eine Pertubation aus der Schnittmenge der Adv-SS gefunden werden. Es kann dabei angenommen werden, dass sich durch den Einsatz eines diversen Set an Klassifikatoren die Schnittmenge an möglichen Pertubationen zur Generierung eines Adversarial Examples reduzieren lässt. Dadurch wird es schwerer für den Angreifer, ein Adversarial Example x' zu finden, dass die Anwendung zur Missklassifikation führt. Dies wird in Abbildung 5.5 veranschaulicht. Figure 1 (links) zeigt den Raum an möglichen Pertubationen, wobei die blau dargestellte Menge die Pertubationen umfasst, die eine fehlerhafte Prädiktion bezwecken. Figure 2 (mitte) zeigt, dass durch die Kombination von Klassifikatoren ein neuer Subspace an schadhaften Pertubationen entsteht. Damit eine fehlerhafte Prädiktion entsteht, müssen damit mehrere Klassifikatoren getäuscht werden. Wie in Figure 3 (rechts) dargestellt, kann ein Ensemble durch die Diversifizierung die Adversarial Robustness maximieren. Wird ein diverses Ensemble zur Klassifikation eingesetzt, reduziert sich die Schnittmenge an Adv-SS. [28]

Auch die durchgeführten Versuche lassen sich nach dieser Hypothese begründen. Da der Angreifer nur die Möglichkeit hat, ein Adversarial Example aus dem Adv-SS eines Klassifikators zu finden, hängt der Erfolg von der Transferabilität der Pertubation ab. Für einen erfolgreichen Angriff muss die Pertubation auch in den Adv SS der anderen Klassifikatoren liegen. Ist dies nicht der Fall, hat der Angriff keine Auswirkung auf die anderen Klassifikatoren und die Anwendung ist immer noch in der Lage, eine valide Prädiktion abzugeben. Ist die Transferabilität des Angriffes gewährleistet, bietet der Einsatz eines Ensembles keinen Schutz gegenüber dem Adversarial Example x'. Dies ist anschaulich in Abbildung 5.7 zu erkennen. Hier wird ein Adversarial Example anhand der Informationen über die logistische Regression, der Support Vector Machine und dem neuronalen Netz generiert. Die so erzeugte Instanz ist in der Lage, die Mehrheit der Klassifikatoren zu täuschen. Dadurch ergibt sich kein Vorteil durch den Einsatz eines Ensembles, stattdessen kann sogar ein einzelner Klassifikator (CNN) eine bessere Adversarial Robustness aufweisen.

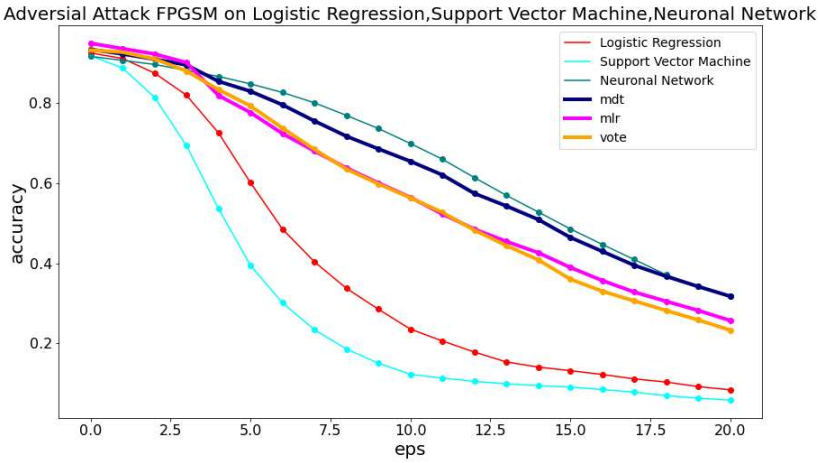


Abbildung 5.7.: Robustheit der Ensemble-Verfahren gegenüber Adversarial Example die mittels FGSM erzeugt. Die Adversarial Examples wurden anhand der Informationen über das Modell der logistischen Regression, Neuronale Netz und der Support Vector Machine erzeugt. Dabei wurde iterativ ϵ erhöht. Als Datensatz wurden die Testdaten des MNIST Digits verwendet

6. Fazit und Ausblick

Es konnte anhand der durchgeführten Versuche bestätigt werden, dass die Zuverlässigkeit von ML-Anwendungen durch den Einsatz von Ensemble Methoden (Voting, Blending) gesteigert werden kann. Für den Erfolg der Kombination der Vorhersagen gibt es dabei im Grunde zwei entscheidende Faktoren:

- **Diversität:** Die verwendeten Basisklassifikatoren sollten so divers und zuverlässig wie möglich sein. Die Fehlklassifikationen der Basisklassifikatoren sollten dabei möglichst unkorreliert miteinander sein. Sie sollen also unterschiedliche Fehler machen. Gerade beim Voting ist es entscheidend, dass für eine Instanz x nicht die Mehrheit an Klassifikatoren denselben Fehler macht. Hat die Mehrheit der Klassifikatoren Schwierigkeiten, bestimmte Objekte fehlerfrei zu klassifizieren, ist ein Votingschema nicht sinnvoll anwendbar. Haben Blending-Verfahren ein sinnvolles Kombinationsschema entwickelt, können sie in solch einer Situation trotzdem das korrekte Label vorhersagen. Zur Kombination der Vorhersagen sollte bevorzugt statt Voting ein Blending-Verfahren eingesetzt werden, da ein geringerer Grad an Diversität benötigt wird.
- **Meta-Learning:** Der zweite entscheidende Faktor ist die Wahl des Kombinationsverfahrens. Beim Voting sollte ein Verfahren verwendet werden, welches die Stärken und Schwächen der Klassifikatoren bei der Entscheidungsfindung berücksichtigt. Das Meta-Learning besteht beim Blending aus zwei Herausforderungen. Es muss ein geeigneter Meta-Learner verwendet werden und eine dazu passende Auswahl an Meta-Features. In den von uns durchgeführten Klassifikationsaufgaben erwies sich der Einsatz der logistischen Regression als Meta-Learner als bestes Verfahren, um die Zuverlässigkeit zu verbessern. Dabei ist anzumerken, dass dies nur für Klassifikationsaufgaben mit mehr als zwei Klassen gilt. Die Robustheit konnte am besten verbessert werden, wenn ein Decision Tree als Meta-Learner verwendet wurde. Die besten Klassifikationsergebnisse konnten erzielt werden, wenn die Meta-Features Informationen über die Konfidenz der Vorhersagen der Basisklassifikatoren enthalten. Für das Blending mittels logistischer Regression zeigten sich Class Probabilities als sinnvolle Meta-Features. Class Probabilities haben den Vorteil, dass mehr Meta-Features zur Verfügung stehen. Des Weiteren konnte beobachtet werden, dass die logistische Regression besser mit kontinuierlichen Variablen arbeitet. Für Blending mittels Decision Tree konnte mit diskreten Labels und

der Entropie der Class Probabilities die besten Ergebnisse erzielt werden. Allgemein sollten jedoch beide Meta-Learner ausprobiert werden, da die Performance stark von der jeweiligen Klassifikationsaufgabe abhängt.

Um Blending Verfahren zu optimieren, sollten Verfahren zur Maximierung der Diversität der Basisklassifikatoren untersucht werden. So könnte eine Feature Selection oder Bootstrapping dafür eingesetzt werden, die Diversität zu erhöhen. Je diversere Basisklassifikatoren eingesetzt werden, desto größer ist das Potenzial der möglichen Optimierung. Es ist auch interessant, weitere Klassifikationsverfahren auf ihre Eignung als Meta-Learner zu untersuchen. Zudem sollte analysiert werden, welche wichtigen Informationen sich noch aus den Vorhersagen der Basisklassifikatoren oder den Inputdaten ableiten lassen und als Meta-Features genutzt werden können. So könnte ein interessanter Ansatz sein, klassenunabhängige Eigenschaften der Daten zu erfassen, welche Einfluss auf die Güte der Klassifikation haben.

Auch die Robustheit der Anwendung ließ sich in den durchgeführten Versuchen durch den Einsatz von Ensemble-Verfahren verbessern. Mithilfe von Votingschemata können redundante Systeme erzeugt werden. Diese können valide Prädiktionen liefern, solange die auftretende Störung den Klassifikationsprozess der Mehrheit der Klassifikatoren nicht nachteilig beeinflusst. Auch die durch das Meta-Learning entwickelten Kombinationsverfahren weisen ein deutlich robusteres Verhalten auf als ein einzelner Klassifikator. Besonders bei der Adversarial Robustness ist das Potenzial von Ensemble Methoden deutlich geworden. Durch den Einsatz möglichst diverser Klassifikatoren kann der Raum an möglichen Pertubationen, die zur Missklassifikation führen, reduziert werden. Ensemble Methoden stellen eine interessante Ergänzung zum Adversarial Training und der Noise Detection dar. Es sollte untersucht werden, ob durch Adversarial Training einzelner Basisklassifikatoren die Schnittmenge der ADV-SS nochmals reduziert werden kann. Somit könnte der Aufwand für den Angreifer, Adversielle Pertubationen zu finden, erhöht werden. Um robuste Anwendungen zu erschaffen, könnten Blending Verfahren zur Classifier Selection ein geeignetes Verfahren darstellen. Es gilt zu untersuchen, ob mittels Fault Injections Ensemble-Verfahren erzeugt werden können, die "wissen", bei welcher Störung welche Basisklassifikatoren genutzt werden sollten.

A. Literaturverzeichnis

[1] Benso Alfredo und Prinetto Paolo. *Fault injection techniques and tools for embedded systems reliability evaluation*. Bd. 23. Frontiers in electronic testing. Boston: Kluwer Academic Publishers, 2003. ISBN: 978-0-306-48711-8.

[2] Klaus Backhaus. *Multivariate Analysemethoden: Eine anwendungsorientierte Einführung*. 14. Aufl. 2016. Springer eBook Collection. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016. ISBN: 9783662460764. DOI: 10.1007/978-3-662-46076-4.

[3] Klaus Backhaus, Bernd Erichson und Rolf Weiber. *Fortgeschrittene multivariate Analysemethoden: Eine anwendungsorientierte Einführung*. 3., überarbeitete und aktualisierte Auflage. Berlin und Heidelberg: Springer Berlin Heidelberg, 2015. ISBN: 9783662460863. DOI: 10.1007/978-3-662-46087-0.

[4] Alejandro Baldominos, Yago Saez und Pedro Isasi. „A Survey of Handwritten Character Recognition with MNIST and EMNIST“. In: *Applied Sciences* 9.15 (2019), S. 3169. DOI: 10.3390/app9153169.

[5] Ali Behravan u. a. *Fault injection framework for fault diagnosis based on machine learning in heating and demand-controlled ventilation systems*. 2017. DOI: 10.1109/KBEI.2017.8324986.

[6] Anup Bhande. *What is underfitting and overfitting in machine learning and how to deal with it*. Aufgerufen am 18.04.2022. URL: https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76.

[7] Christopher M. Bishop. *Pattern recognition and machine learning*. Information science and statistics. New York: Springer, 2006. ISBN: 0387310738.

[8] Leo Breiman. „Random Forests“. In: *Machine Learning* 45.1 (2001), S. 5–32. ISSN: 0885-6125. DOI: 10.1023/A:1010933404324.

[9] Leo Breiman u. a. *Classification And Regression Trees*. Routledge, 2017. DOI: 10.1201/9781315139470.

[10] Erica Briscoe und Jacob Feldman. „Conceptual complexity and the bias/variance tradeoff“. In: *Cognition* 118.1 (2011), S. 2–16. DOI: 10.1016/j.cognition.2010.10.004.

[11] Peter Buxmann und Holger Schmidt, Hrsg. *Künstliche Intelligenz: Mit Algorithmen zum wirtschaftlichen Erfolg*. 2., aktualisierte und erweiterte Auflage. Berlin, Heidelberg: Springer Berlin Heidelberg, 2021. ISBN: 9783662617939. DOI: 10.1007/978-3-662-61794-6.

[12] Xi Chen und Hemant Ishwaran. „Random forests for genomic data analysis“. In: *Genomics* 99.6 (2012), S. 323–329. DOI: 10.1016/j.ygeno.2012.04.003.

[13] Wiesław Chmielnicki und Katarzyna Stąpor. „Using the one–versus–rest strategy with samples balancing to improve pairwise coupling classification“. In: *International Journal of Applied Mathematics and Computer Science* 26.1 (2016), S. 191–201. ISSN: 1641-876X. DOI: 10.1515/amcs-2016-0013.

[14] Brune Christoph und Burger Martin. „Grundaufgaben der Bildverarbeitung“. In: Variationsmethoden in der Biomedizinischen Bildgebung - Vorlesungsskript (WS 2010/2011).

[15] Thomas G. Dietterich. *Ensemble Methods in Machine Learning*. Bd. 1857. Lecture Notes in Computer Science - Multiple Classifier Systems. Springer Berlin Heidelberg, 2000, S. 1–15. ISBN: 978-3-540-67704-8. DOI: 10.1007/3-540-45014-9_1.

[16] Tom Dietterich. „Overfitting and undercomputing in machine learning“. In: *ACM Computing Surveys* 27.3 (1995), S. 326–327. ISSN: 0360-0300. DOI: 10.1145/212094.212114.

[17] Richard O. Duda, Peter E. Hart und David G. Stork. „Pattern classification“. In: second edition. Wiley-Interscience, 2001. ISBN: 0-471-05669-3.

[18] Saso Džeroski und Bernard Ženko. „Is Combining Classifiers with Stacking Better than Selecting the Best One?“ In: *Machine Learning* 54.3 (2004), S. 255–273. ISSN: 0885-6125. DOI: 10.1023/B:MACH.0000015881.36452.6e.

[19] Fabian Pedregosa u. a. „Scikit-learn: Machine Learning in Python“. In: *Journal of Machine Learning Research* 12.85 (2011), S. 2825–2830. ISSN: 1533-7928. DOI: 10.48550/arXiv.1201.0490.

[20] Matthias Franz. „Mustererkennung und Klassifikation - Einführung“. In: (WS 2007/08). URL: http://www-home.fh-konstanz.de/~mfranz/muk_files/lect00_Einleitung.pdf.

[21] Frederick Kazman u. a. *Robustness*. 2022. DOI: 10.1184/R1/16455660.v1.

[22] Jerome H. Friedman. „On Bias, Variance, 0/1—Loss, and the Curse-of-Dimensionality“. In: *Data Mining and Knowledge Discovery* 1.1 (1997), S. 55–77. ISSN: 13845810. DOI: 10.1023/A:1009778005914.

[23] Pascal Getreuer. „A Survey of Gaussian Convolution Algorithms“. In: *Image Processing On Line* 3 (2013), S. 286–310. DOI: 10.5201/ipol.2013.87.

[24] Ian J. Goodfellow, Jonathon Shlens und Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2014. DOI: 10.48550/arXiv.1412.6572.

[25] Chuan Guo u. a. *Simple Black-box Adversarial Attacks*. 2019. DOI: 10.48550/arXiv.1905.07121.

[26] R. A. Haddad und A. N. Akansu. „A class of fast Gaussian binomial filters for speech and image processing“. In: *IEEE Transactions on Signal Processing* 39.3 (1991), S. 723–727. ISSN: 1053587X. DOI: 10.1109/78.80892.

[27] Daniel Jacobi und Gunther Hellmann, Hrsg. *Das Weißbuch 2016 und die Herausforderungen von Strategiebildung: Zwischen Notwendigkeit und Möglichkeit*. Edition ZfAS. Wiesbaden: Springer VS, 2019. ISBN: 9783658239749.

[28] Sanjay Kariyappa und Moinuddin K. Qureshi. *Improving Adversarial Robustness of Ensembles with Diversity Training*. 2019. DOI: 10.48550/arXiv.1901.09981.

[29] Hojat Khosrowjerdi, Karl Meinke und Andreas Rasmusson. „Virtualized-Fault Injection Testing: A Machine Learning Approach“. In: *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2018, S. 297–308. ISBN: 978-1-5386-5012-7. DOI: 10.1109/ICST.2018.00037.

[30] Tobias D. Krafft. *Qualitätsmaße binärer Klassifikationen im Bereich kriminalprognostischer Instrumente der vierten Generation*. 2018. DOI: 10.48550/arXiv.1804.01557.

[31] Ludmila I. Kuncheva. *Combining pattern classifiers: Methods and algorithms*. Hoboken NJ: J. Wiley, 2004. ISBN: 0471210781.

[32] Guanpeng Li, Karthik Pattabiraman und Nathan DeBardeleben. „TensorFI: A Configurable Fault Injector for TensorFlow Applications“. In: *2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)* (2018), S. 313–320. DOI: 10.1109/ISSREW.2018.00024.

[33] Wei Liu und Weisi Lin. „Additive white Gaussian noise level estimation in SVD domain for images“. In: *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society* 22.3 (2013), S. 872–883. DOI: 10.1109/TIP.2012.2219544.

[34] Pankaj Mehta u. a. „A high-bias, low-variance introduction to Machine Learning for physicists“. In: *Physics reports* 810 (2019), S. 1–124. ISSN: 0370-1573. DOI: 10.1016/j.physrep.2019.03.001.

[35] Anthony J. Myles u. a. „An introduction to decision tree modeling“. In: *Journal of Chemometrics* 18.6 (2004), S. 275–285. ISSN: 0886-9383. DOI: 10.1002/cem.873.

[36] Niranjhana Narayanan. „Fault injection in Machine Learning applications“. Diss. 2021. DOI: 10.14288/1.0396949.

[37] Brady Neal u. a. *A Modern Take on the Bias-Variance Tradeoff in Neural Networks*. 2018. DOI: 10.48550/ARXIV.1810.08591.

[38] Praneeth Netrapalli. „Stochastic Gradient Descent and Its Variants in Machine Learning“. In: *Journal of the Indian Institute of Science* 99.2 (2019), S. 201–213. ISSN: 0970-4140. DOI: 10.1007/s41745-019-0098-4.

[39] Maria-Irina Nicolae u. a. *Adversarial Robustness Toolbox v1.0.0*. 2018. DOI: 10.48550/ARXIV.1807.01069.

[40] Gerhard Paass und Dirk Hecker. *Künstliche Intelligenz: Was steckt hinter der Technologie der Zukunft?* Wiesbaden und Heidelberg: Springer Fachmedien Wiesbaden, 2020. ISBN: 978-3-658-30210-8. DOI: 10.1007/978-3-658-30211-5.

[41] Mario A. Pfannstiel, Hrsg. *Künstliche Intelligenz im Gesundheitswesen: Entwicklungen, Beispiele und Perspektiven*. Wiesbaden, Germany: Springer Gabler, 2022. ISBN: 9783658335960. DOI: 10.1007/978-3-658-33597-7.

[42] Stefan Richter. *Statistisches und maschinelles Lernen*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2019. ISBN: 978-3-662-59353-0. DOI: 10.1007/978-3-662-59354-7.

[43] Suchi Saria und Adarsh Subbaswamy. *Tutorial: Safe and Reliable Machine Learning*. 2019. DOI: 10.48550/ARXIV.1904.07204.

[44] David B. Skalak. „The Sources of Increased Accuracy for Two Proposed Boosting Algorithms“. In: *In Proc. American Association for Arti Intelligence, AAAI-96, Integrating Multiple Learned Models Workshop*. 1996, S. 120–125.

[45] Marina Sokolova und Guy Lapalme. „A systematic analysis of performance measures for classification tasks“. In: *Information Processing Management* 45.4 (2009), S. 427–437. ISSN: 0306-4573. DOI: 10.1016/j.ipm.2009.03.002.

[46] Priyanka Gupta Sonia Singh. *Comparative study ID3, cart and C4. 5 decision tree algorithm: a survey*. 2014.

[47] Daniel Sonntag. „Künstliche Intelligenz in der Medizin – Holzweg oder Heilversprechen?“ In: *HNO* 67.5 (2019), S. 343–349. DOI: 10.1007/s00106-019-0665-z.

[48] „Stacked generalization: when does it work?“ In: *1170-487X* (1997), S. 866–871. ISSN: 1170-487X.

[49] Thilo Strauss u. a. *Ensemble Methods as a Defense to Adversarial Perturbations Against Deep Neural Networks*. 2017. DOI: 10.48550/ARXIV.1709.03423.

[50] Kanae Takahashi u. a. „Confidence interval for micro-averaged F1 and macro-averaged F1 scores“. In: *Applied intelligence (Dordrecht, Netherlands)* 52.5 (2022), S. 4961–4972. DOI: 10.1007/s10489-021-02635-5.

[51] Eun Bae Kong Thomas G. Dietterich. *Machine Learning Bias, Statistical Bias, and Statistical Variance of Decision Tree Algorithms*. 1995.

[52] K. M. Ting und I. H. Witten. „Issues in Stacked Generalization“. In: *Journal of Artificial Intelligence Research* 10 (1999), S. 271–289. ISSN: 1076-9757. DOI: 10.1613/jair.594.

[53] Ljupco Todorovski und Sašo Džeroski. „Combining Classifiers with Meta Decision Trees“. In: *Machine Learning* 50 (2003). DOI: 10.1023/A:1021709817809.

[54] K.K.V. Toh und N.A.M. Isa. „Noise Adaptive Fuzzy Switching Median Filter for Salt-and-Pepper Noise Reduction“. In: *IEEE Signal Processing Letters* 17.3 (2010), S. 281–284. ISSN: 1070-9908. DOI: 10.1109/LSP.2009.2038769.

[55] Yoshimasa Tsuruoka, Jun'ichi Tsujii und Sophia Ananiadou. „Stochastic gradient descent training for L1-regularized log-linear models with cumulative penalty“. In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*. Hrsg. von Keh-Yih Su. ACM Digital Library. United States: Association for Computational Linguistics, 2009, S. 477. ISBN: 9781932432459. DOI: 10.3115/1687878.1687946.

[56] Abhisek Ukil. *Intelligent Systems and Signal Processing in Power Engineering*. Power Systems. Springer Berlin Heidelberg, 2007. ISBN: 9783540731702. DOI: 10.1007/978-3-540-73170-2.

[57] Martin Werner. *Digitale Bildverarbeitung*. Springer Fachmedien Wiesbaden, 2021. ISBN: 978-3-658-22184-3. DOI: 10.1007/978-3-658-22185-0.

[58] Thomas Westermeier. „Multi-Class Classification: Vergleich von Support Vector Machine mit anderen ausgewählten Algorithmen“. Diss. Universitätsbibliothek der Ludwig-Maximilians-Universität München, 2019. DOI: 10.5282/ubm/epub.68478.

[59] David H. Wolpert. „Stacked generalization“. In: *Neural Networks* 5.2 (1992), S. 241–259. ISSN: 08936080. DOI: 10.1016/S0893-6080(05)80023-1.

[60] Laurenz Wuttke. *Machine Learning: Definition, Algorithmen, Methoden und Beispiele*. Aufgerufen am 18.04.2022. URL: https://datasolut.com/was-ist-machine-learning/.

[61] Han Xiao, Kashif Rasul und Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. 2017. DOI: 10.48550/ARXIV.1708.07747.

[62] Wei Xu. *Towards Optimal One Pass Large Scale Learning with Averaged Stochastic Gradient Descent*. 2011. DOI: 10.48550/ARXIV.1107.2490.

[63] Jiancheng Yang u. a. *MedMNIST v2: A Large-Scale Lightweight Benchmark for 2D and 3D Biomedical Image Classification*. 2021. DOI: 10.48550/ARXIV.2110.14795.

[64] Zhi-Hua Zhou. *Ensemble methods: Foundations and algorithms*. Online-Ausg. Chapman & Hall. Boca Raton, FL: Taylor & Francis, 2012. ISBN: 9781439830055.

[65] Ziad Alqadi. „Salt and Pepper Noise: Effects and Removal“. In: *International Journal on Electrical Engineering and Informatics* 2.4 (2018). ISSN: 2085-6830.

B. Anhang

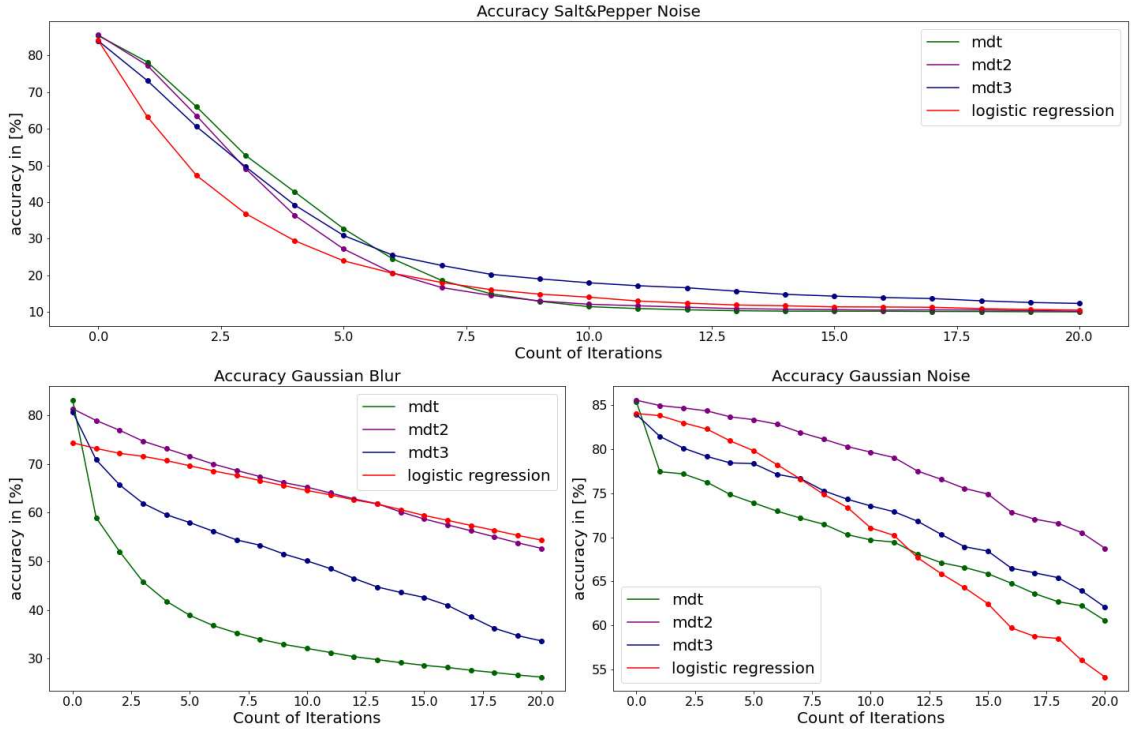


Abbildung B.1.: Robustheit Voting-Verfahren für Salt & Pepper Noise(a), Gaussian Blur(b) und Gaussian Noise(c). Dabei wurde iterativ die Stärke der Störungen erhöht. Als Datensatz wurden die Testdaten des Fashion MNIST verwendet.

Tabelle B.1.: Klassifikationsergebnisder Basisklassifikatoren für Validationsdaten des MNIST Digits Datensatzes

Algorithmus	accuracy	precision	recall	f1-measure
Logistic Regression	**91.99**	**91.90**	91.91	**91.90**
Stochastic Gradient Descent	91.05	90.94	91.06	90.91
Decision Tree	86.63	86.45	86.48	86.46
Random Forest	90.55	90.43	90.66	90.47
Convolutional Neural Networks	91.42	91.31	**92.34**	91.53
Support Vector Machine	91.24	91.13	91.12	91.12
Linear Discriminant Analysis	86.47	86.37	86.59	86.37

Tabelle B.2.: Klassifikationsergebnisder Basisklassifikatoren für Validationsdaten des MNIST Digits Datensatzes

Algorithmus	accuracy	precision	recall	f1-measure
Logistic Regression	92.39	92.28	92.30	92.28
Stochastic Gradient Descent	91.15	91.02	91.14	90.99
Decision Tree	87.61	87.43	87.46	87.44
Random Forest	91.34	91.20	91.39	91.24
Convolutional Neural Networks	91.50	91.36	92.39	91.55
Support Vector Machine	91.66	91.54	91.54	91.53
Linear Discriminant Analysis	87.25	87.11	87.33	87.12
Voting	93.03	92.94	92.96	92.93
Meta Decision tree	92.78	92.68	92.72	92.69
Meta Decision tree 2	92.71	92.61	92.62	92.60
Meta Decision tree 3	92.93	92.84	92.87	92.85
Meta Logistic Regression	**94.71**	**94.64**	**94.66**	**94.65**
Meta Logistic Regression2	86.30	86.09	86.16	86.06
Meta Logistic Regression3	**77.35**	**76.92**	**76.99**	**76.88**

Tabelle B.3.: Klassifikationsergebnisder Basisklassifikatoren für Validationsdaten des MNIST Fashion Datensatzes

Algorithmus	accuracy	precision	recall	f1-measure
Logistic Regression	**85.12**	**85.17**	85.07	**85.09**
Stochastic Gradient Descent	84.63	84.67	84.97	84.71
Decision Tree	79.80	79.84	79.87	79.85
Random Forest	83.13	83.21	83.36	82.83
Convolutional Neural Networks	84.28	84.34	**86.13**	84.42
Support Vector Machine	84.79	84.85	84.65	84.66
Linear Discriminant Analysis	82.23	82.26	82.51	82.35

Tabelle B.4.: Klassifikationsergebnisder Basisklassifikatoren für Testdaten des MNIST Fashion Datensatzes

Algorithmus	accuracy	precision	recall	f1-measure
Logistic Regression	84.05	84.05	83.94	83.97
Stochastic Gradient Descent	83.36	83.36	83.68	83.42
Decision Tree	78.98	78.98	79.07	79.02
Random Forest	82.17	82.17	82.36	81.80
Convolutional Neural Networks	83.35	83.35	85.19	83.43
Support Vector Machine	83.96	83.96	83.72	83.75
Linear Discriminant Analysis	81.33	81.33	81.53	81.40
Voting	85.32	85.32	85.21	85.22
Meta Decision tree	85.15	85.15	85.36	85.20
Meta Decision tree 2	84.98	84.98	84.95	84.93
Meta Decision tree 3	83.15	83.15	82.98	83.03
Meta Logistic Regression	**86.57**	**86.57**	**86.48**	**86.50**
Meta Logistic Regression2	75.38	75.38	74.56	74.37
Meta Logistic Regression3	73.35	73.35	73.41	73.32

Tabelle B.5.: Klassifikationsergebnisder Basisklassifikatoren für Validationsdaten des MedMNIST OrganA Datensatzes

Algorithmus	accuracy	precision	recall	f1-measure
Logistic Regression	79.22	75.35	73.57	74.45
Stochastic Gradient Descent	74.21	70.68	71.09	70.89
Decision Tree	78.26	77.38	77.81	77.59
Random Forest	71.58	66.04	65.17	65.60
Convolutional Neural Networks	**84.55**	80.90	81.14	81.02
Support Vector Machine	71.47	61.06	62.87	61.95
Linear Discriminant Analysis	83.50	**81.01**	**82.33**	**81.66**

Tabelle B.6.: Klassifikationsergebnisder Basisklassifikatoren für Testdaten des MedMNIST OrganA Datensatzes

Algorithmus	accuracy	precision	recall	f1-measure
Logistic Regression	55.06	54.87	52.98	53.91
Stochastic Gradient Descent	52.89	51.24	54.78	52.95
Decision Tree	60.07	59.92	61.69	60.80
Random Forest	58.14	53.28	59.49	56.22
Convolutional Neural Networks	63.35	61.04	63.74	62.36
Support Vector Machine	53.27	47.61	52.04	49.73
Linear Discriminant Analysis	58.65	56.44	64.12	60.04
Voting	50.75	67.72	50.64	57.95
Meta Decision tree	61.33	60.59	65.58	62.99
Meta Decision tree 2	63.68	62.29	65.33	63.77
Meta Decision tree 3	60.86	60.81	64.01	62.37
Meta Logistic Regression	**67.07**	**66.28**	**71.49**	**68.79**
Meta Logistic Regression2	51.57	47.83	52.24	49.94
Meta Logistic Regression3	54.26	54.62	60.68	57.49

Tabelle B.7.: Klassifikationsergebnisder Basisklassifikatoren für Validationsdaten des MedMNIST OrganC Datensatzes

Algorithmus	accuracy	precision	recall	f1-measure
Logistic Regression	73.12	69.96	68.55	69.24
Stochastic Gradient Descent	63.29	59.09	71.91	64.87
Decision Tree	76.05	73.85	72.99	73.42
Random Forest	78.68	73.92	72.62	73.27
Convolutional Neural Networks	74.96	72.19	76.82	74.44
Support Vector Machine	69.65	62.99	64.97	63.97
Linear Discriminant Analysis	**81.31**	**81.12**	**79.70**	**80.40**

Tabelle B.8.: Klassifikationsergebnisder Basisklassifikatoren für Testdaten des MedMNIST OrganC Datensatzes

Algorithmus	accuracy	precision	recall	f1-measure
Logistic Regression	57.09	57.05	55.60	56.31
Stochastic Gradient Descent	48.33	48.91	60.64	54.15
Decision Tree	64.79	63.39	64.59	63.99
Random Forest	66.51	63.18	71.77	67.20
Convolutional Neural Networks	61.10	58.91	68.55	63.37
Support Vector Machine	55.58	51.74	54.54	53.10
Linear Discriminant Analysis	64.73	62.45	68.14	65.17
Voting	64.08	**70.17**	62.11	65.90
Meta Decision tree	67.90	65.76	72.24	68.84
Meta Decision tree 2	68.69	67.92	71.04	69.44
Meta Decision tree 3	67.78	65.70	69.80	67.69
Meta Logistic Regression	**71.14**	69.49	**74.30**	**71.81**
Meta Logistic Regression2	48.73	45.67	44.62	45.14
Meta Logistic Regression3	55.42	53.58	59.30	56.29

Tabelle B.9.: Klassifikationsergebnisder Basisklassifikatoren für Validationsdaten des MedMNIST Breast Datensatzes

Algorithmus	accuracy	precision	recall	f1-measure
Logistic Regression	76.28	**70.24**	69.94	70.09
Stochastic Gradient Descent	78.21	70.05	72.22	**70.94**
Decision Tree	73.72	67.73	67.00	67.33
Random Forest	**80.77**	66.54	**81.88**	69.05
Convolutional Neural Networks	76.92	60.15	74.45	60.83
Support Vector Machine	75.00	65.60	67.64	66.38
Linear Discriminant Analysis	71.79	68.67	66.23	66.90

Tabelle B.10.: Klassifikationsergebnisder Basisklassifikatoren für Testdaten des MedMNIST Breast Datensatzes

Algorithmus	accuracy	precision	recall	f1-measure
Logistic Regression	76.28	69.94	70.24	70.09
Stochastic Gradient Descent	78.21	70.05	72.22	70.94
Decision Tree	73.72	67.73	67.00	67.33
Random Forest	80.77	66.54	81.88	69.05
Convolutional Neural Networks	76.92	60.15	74.45	60.83
Support Vector Machine	75.00	65.60	67.64	66.38
Linear Discriminant Analysis	71.79	68.67	66.23	66.90
Voting	80.13	75.20	71.37	72.79
Meta Decision tree	75.64	74.31	70.79	71.73
Meta Decision tree 2	75.00	65.60	67.64	66.38
Meta Decision tree 3	65.38	68.80	64.80	63.20
Meta Logistic Regression	78.85	71.24	73.10	72.03
Meta Logistic Regression2	78.85	71.99	73.08	72.49
Meta Logistic Regression3	**79.49**	**72.43**	**73.95**	**73.10**

Tabelle B.11.: Klassifikationsergebnisder Basisklassifikatoren für Validationsdaten des MedMNIST Pneumonial Datensatzes

Algorithmus	accuracy	precision	recall	f1-measure
Logistic Regression	81.25	75.94	85.12	77.63
Stochastic Gradient Descent	82.85	77.74	**87.17**	79.61
Decision Tree	79.49	74.62	81.31	76.00
Random Forest	83.49	79.02	86.43	80.77
Convolutional Neural Networks	**84.78**	**81.58**	85.71	**82.89**
Support Vector Machine	82.85	77.82	86.91	79.67
Linear Discriminant Analysis	82.85	77.65	87.43	79.54

Tabelle B.12.: Klassifikationsergebnisder Basisklassifikatoren für Testdaten des MedMNIST Pneumonial Datensatzes

Algorithmus	accuracy	precision	recall	f1-measure
Logistic Regression	81.25	85.12	75.94	77.63
Stochastic Gradient Descent	82.85	77.74	87.17	79.61
Decision Tree	79.49	74.62	81.31	76.00
Random Forest	83.49	79.02	86.43	80.77
Convolutional Neural Networks	84.78	81.58	85.71	82.89
Support Vector Machine	82.85	77.82	86.91	79.67
Linear Discriminant Analysis	82.85	77.65	87.43	79.54
Voting	84.29	**88.82**	79.40	81.40
Meta Decision tree	**86.70**	83.46	**88.34**	**84.98**
Meta Decision tree 2	82.85	78.25	85.77	79.96
Meta Decision tree 3	86.54	83.68	87.49	84.96
Meta Logistic Regression	85.26	81.20	88.04	82.98
Meta Logistic Regression2	85.10	80.98	87.93	82.78
Meta Logistic Regression3	85.74	82.09	87.81	83.73

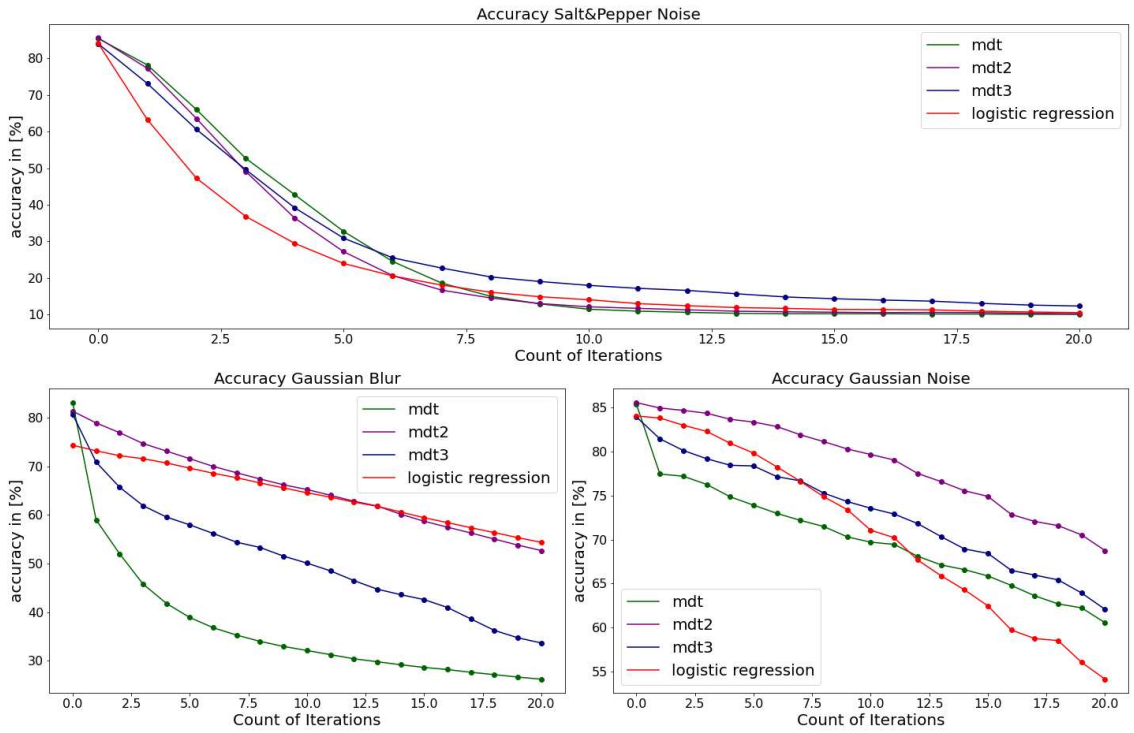


Abbildung B.2.: Robustheit der Blending Verfahren mit Decision Tree als Meta-Learner für Salt & Pepper Noise(a), Gaussian Blur(b) und Gaussian Noise(c). Dabei wurde iterativ die Stärke der Störungen erhöht. Als Datensatz wurden die Testdaten des Fashion MNIST verwendet.

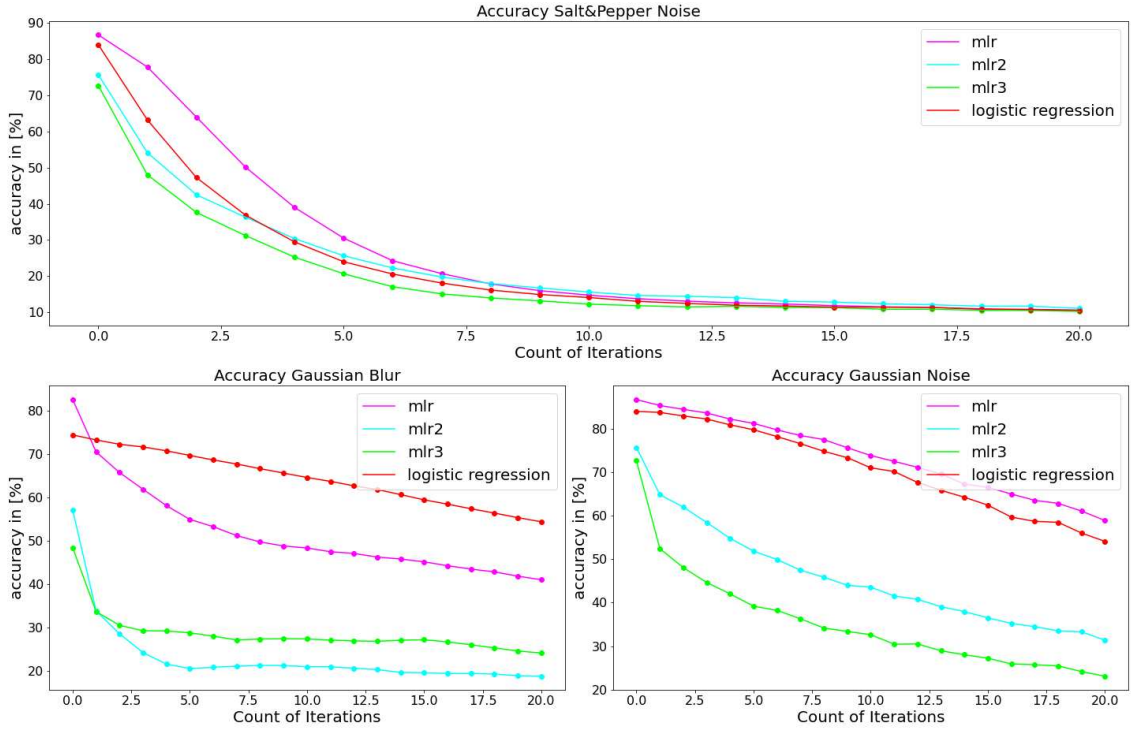


Abbildung B.3.: Robustheit der Blending Verfahren mit logistischer Regression als Meta-Learner für Salt & Pepper Noise(a), Gaussian Blur(b) und Gaussian Noise(c). Dabei wurde iterativ die Stärke der Störungen erhöht. Als Datensatz wurden die Testdaten des Fashion MNIST verwendet.

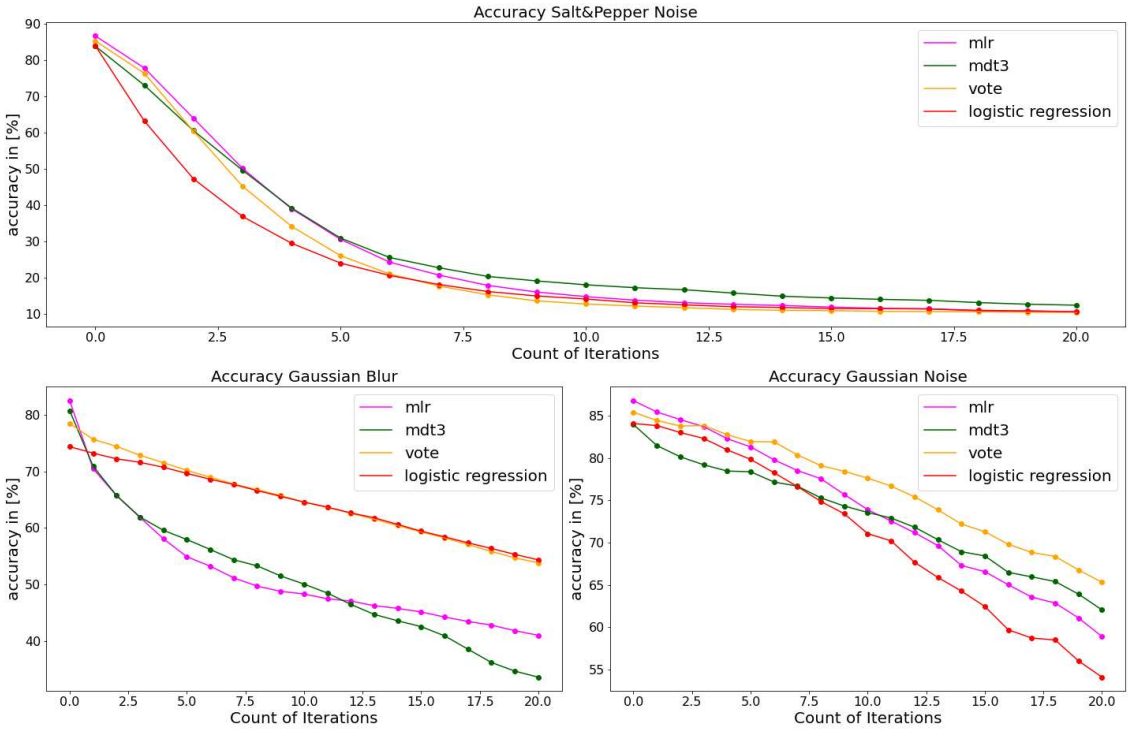


Abbildung B.4.: Robustheit Ensemble-Verfahren für Salt & Pepper Noise(a), Gaussian Blur(b) und Gaussian Noise(c). Dabei wurde iterativ die Stärke der Störungen erhöht. Als Datensatz wurden die Testdaten des Fashion MNIST verwendet.

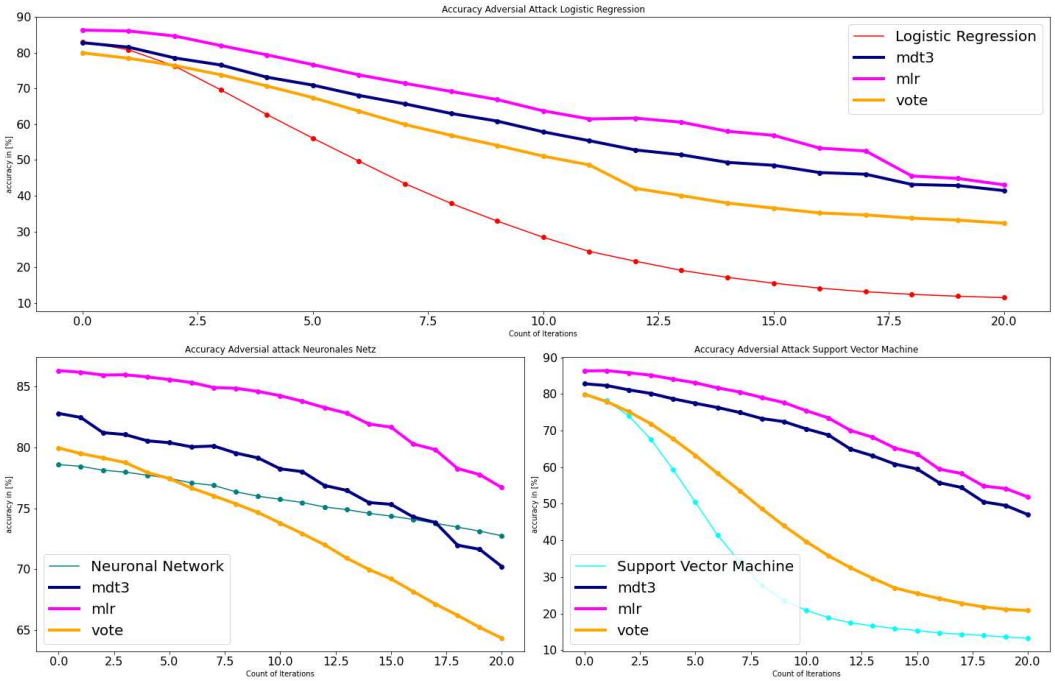


Abbildung B.5.: Robustheit der Ensemble-Verfahren gegenüber Adversarial Example die mittels FGSM erzeugt. Die Adversarial Examples wurden anhand der Informationen über das Modell der logistischen Regression (a), Neuronale Netz (b) und der Support Vector Machine (c) erzeugt. Dabei wurde iterativ ϵ erhöht. Als Datensatz wurden die Testdaten des Fashion MNIST verwendet.

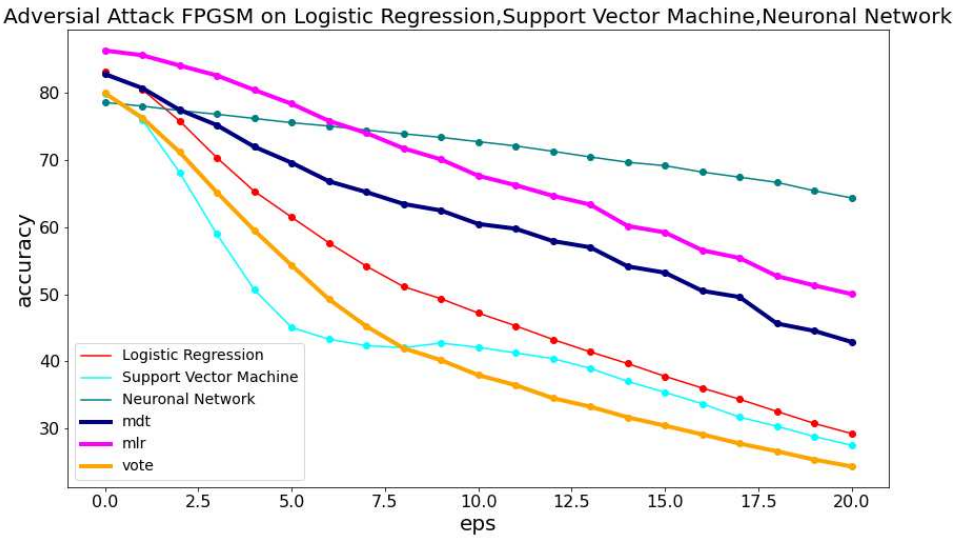


Abbildung B.6.: Robustheit der Ensemble-Verfahren gegenüber Adversarial Example die mittels FGSM erzeugt. Die Adversarial Examples wurden anhand der Informationen über das Modell der logistischen Regression, Neuronale Netz und der Support Vector Machine erzeugt. Dabei wurde iterativ ϵ erhöht. Als Datensatz wurden die Testdaten des Fashion MNIST verwendet.