



OTTO VON GUERICKE
UNIVERSITÄT
MAGDEBURG



FAKULTÄT FÜR
INFORMATIK

Otto-von-Guericke-University Magdeburg

Fakultät für Informatik

Chair of Computational Intelligence

Vergleich ausgewählter Methoden zur
Uncertainty Estimation und deren
Verwendung für aktives Lernen

Bachelorarbeit

Autorin:

Hanna Lichtenberg

Prüferin:

Prof. Dr.-Ing. habil. Sanaz Mostaghim

Zweitprüfer:

Dr.-Ing. Christoph Steup

Betreuer:

Sören Möllering

Magdeburg, 05.04.2023

Kurzfassung

Durch viele Fortschritte im Bereich der künstlichen neuronalen Netzwerke besteht das Bestreben, diese auch in kritischen Situationen einzusetzen. Dazu ist Vertrauen in die Ausgaben der Netzwerke notwendig. Auch eine hohe Genauigkeit der Netzwerke ist ein wichtiger Faktor. Die Abschätzung der prädiktiven Unsicherheit, die *Uncertainty Estimation*, ermöglicht es, Vorhersagen über die Korrektheit von Modellausgaben zu treffen. Somit wird das Vertrauen in viele Ausgaben erhöht. Die prädiktive Unsicherheit kann ebenfalls genutzt werden, um die Effizienz des Labelings als Teil des Trainingsprozesses eines Netzwerks zu erhöhen, sodass dessen Genauigkeit schneller wächst. Eine solche Vorgehensweise zur Optimierung des Trainingsprozesses ist das *unsicherheitsbasierte aktive Lernen*.

In dieser Arbeit werden ausgewählte Methoden zur Abschätzung der prädiktiven Unsicherheit anhand verschiedener Qualitätskriterien verglichen und herausgefunden, wo die Stärken der einzelnen Methoden liegen. Die Tests fokussieren sich unter anderem auf die Interpretierbarkeit und den Informationsgehalt der Uncertainty Estimates sowie auf die Laufzeit der Methoden. Sie werden für die Datensätze Cifar10, Cifar100 und Cars durchgeführt. Die Performance der Methoden für das unsicherheitsbasierte aktive Lernen wird gesondert untersucht. Dabei wird bewertet, wie groß die Verbesserung des Trainingsprozesses durch das aktive Lernen ist. Ziel ist außerdem, herauszufinden, welche Uncertainty Estimation Methoden am besten für das aktive Lernen geeignet sind. Es folgt die Untersuchung einer Erweiterung des unsicherheitsbasierten aktiven Lernens, die verspricht, die Genauigkeit eines Netzwerks noch schneller zu erhöhen. Inwiefern der aktuelle Forschungsstand der Uncertainty Estimation und des aktiven Lernens für die Praxis relevant ist, wird kritisch evaluiert.

Abstract

Due to many advances in the field of artificial neural networks, there is a desire to use them in critical situations. For this, trust in model outputs is necessary. High accuracy of the networks is also an important factor. Estimating predictive uncertainty, known as *Uncertainty Estimation*, enables predictions about the correctness of outputs, thereby increasing confidence in the model outputs. Predictive uncertainty can also be used to improve the efficiency of labeling as part of the training process of a network, so that its accuracy grows faster. One such approach to optimize the training is the *Uncertainty-Based Active Learning*.

In this thesis, selected methods for estimating predictive uncertainty are compared based on various quality criteria, and the strengths of each method are identified. The tests focus on the interpretability and the information content of uncertainty estimates as well as the runtime of the methods, among other factors. The tests are conducted on the Cifar10, Cifar100, and Cars datasets. The performance of the methods for Uncertainty-Based Active Learning is separately evaluated to determine to which extent the Active Learning improves the training process. The goal is also to determine which Uncertainty Estimation methods are best suited for Active Learning. The thesis then examines an extension of Uncertainty-Based Active Learning that promises to further increase a network's accuracy faster. Finally, the current state of research on Uncertainty Estimation and Active Learning is critically evaluated for its practical relevance.

Inhaltsverzeichnis

Notationsverzeichnis	vii
Abkürzungsverzeichnis	ix
1 Einleitung	1
1.1 Motivation	1
1.2 Zielsetzung	2
1.3 Gliederung der Arbeit	3
1.4 Verwandte Arbeiten	3
2 Grundlagen	5
2.1 Künstliche neuronale Netzwerke	5
2.1.1 Aufbau von Deep Feedforward Networks	5
2.1.2 Training künstlicher neuronaler Netzwerke	7
2.1.3 Grundlagen Convolutional Neural Networks (CNNs)	11
2.2 Prädiktive Unsicherheit in DNNs	12
2.2.1 Definition der prädiktiven Unsicherheit eines DNNs	12
2.2.2 Hauptursachen für prädiktive Unsicherheit	13
2.2.3 Klassifizierung der prädiktiven Unsicherheit	14
2.2.4 Zu selbstsichere Softmaxscores für Klassifizierungsprobleme	16
2.3 Methoden zur Uncertainty Estimation in DNNs	17
2.3.1 Überblick und Einteilung von Methoden zur Uncertainty Estimation	17
2.3.2 Bayessche neuronale Netzwerke (BNNs)	18
2.3.3 Monte Carlo Dropout Methode	21
2.3.4 Ensemble Methoden	25
2.3.5 Neighborhood Uncertainty Classifier	28
2.4 Messung der prädiktiven Unsicherheit bei Klassifizierungsproblemen	30
3 Bewertung der Uncertainty Estimation bei Klassifizierungsproblemen	35
3.1 Verwendete Datensätze und Modelle	35
3.2 Umsetzungsdetails der betrachteten Methoden	38
3.3 Evaluierungstests der Methoden zur Uncertainty Estimation	38
3.3.1 Kalibrierungstest - Interpretierbarkeit der Uncertainty Estimates . .	38
3.3.2 Ergebnisse ausgewählter Evaluierungsmetriken	47
3.3.3 Out-of-Domain Erkennung	57
3.3.4 Laufzeitvergleich der Methoden	60

4	Verwendung von Uncertainty Estimates für das aktive Lernen	65
4.1	Skizzierter Ablauf des aktiven Lernens mit Uncertainty Estimates	65
4.2	Bewertung des unsicherheitsbasierten DeepAL am Beispiel von Cifar10 . .	68
4.3	Hybride Auswahlstrategie für das DeepAL	74
4.3.1	Ansatz zur Erhöhung der Diversität ausgewählter Samples	74
4.3.2	Anwendung der hybriden Auswahlstrategie für Cifar10	76
5	Fazit und Ausblick	79
5.1	Zusammenfassung und Auswertung der Erkenntnisse	79
5.2	Uncertainty Estimation für semantische Segmentierung	81
5.3	Aktuelle Probleme und nötige Verbesserungen der Uncertainty Estimation	83
5.4	Ausblick des DeepALs und dessen zukünftige Verwendung bei Volkswagen	85
	Anhang	87
A.	Tabellen	87
B.	Calibration Plots	88
C.	ROC- und PR-Kurven	91
D.	Precision, Specificity und Recall der kalibrierten Uncertainty Estimates . .	93
	Literaturverzeichnis	95
	Selbstständigkeitserklärung	99

Notationsverzeichnis

Zahlen und Felder

\mathbf{A}	Eine Matrix
\mathbf{a}	Ein Vektor
a	Ein Skalar
a_i	Die i -te Komponente des Vektors \mathbf{a}
a_{ij}	Das spezifische Element in der Matrix \mathbf{A} (Reihe i , Spalte j)
$\text{diag}(\mathbf{a})$	Die Diagonalmatrix, deren diagonalen Einträge durch \mathbf{a} gegeben sind
$\mathbf{0}$	Ein Vektor, der lediglich Nullen enthält
\mathbf{A}^T	Die transponierte Matrix \mathbf{A}
\mathbf{I}	Eine Einheitsmatrix
\mathbf{W}	Eine Gewichtsmatrix
\mathbf{b}	Ein Biasvektor

Mengen

\mathbb{R}	Die Menge der reellen Zahlen
\mathbb{N}_0	Die Menge bestehend aus den natürlichen Zahlen und der 0
$\{0, 1\}$	Die Menge, die 0 und 1 enthält
$[a, b]$	Das reelle Intervall einschließlich a und b
(a, b)	Das reelle Intervall ohne a und b

Funktionen

$f_\omega : \mathcal{X} \rightarrow \mathcal{Y}$	Eine Funktion parametrisiert durch ω , die eine Eingabe aus der Domäne \mathcal{X} auf eine Ausgabe der Domäne \mathcal{Y} abbildet
$f_\omega(\mathbf{x})$ oder $f(\mathbf{x}; \omega)$	Die deterministische Prädiktion für \mathbf{x} mit den Gewichten ω
f^*	Die Funktion, die durch f_ω approximiert wird
$\ \mathbf{x}\ $	Die euklidische Länge des Vektors \mathbf{x}
$\log(x)$	Der natürliche Logarithmus von x
$\nabla L(\omega)$	Der Gradient von $L(\omega)$
$\mathbb{1}_{\text{Bedingung}}$	1 falls die Bedingung wahr ist, sonst 0
$\int f(\mathbf{x})d\mathbf{x}$	Ein unbestimmtes Integral über die gesamte Domäne von \mathbf{x}
$\int_A f(\mathbf{x})d\mathbf{x}$	Ein bestimmtes Integral bezüglich \mathbf{x} über alle Elemente in A

Wahrscheinlichkeitstheorie

$p(a)$	Eine Wahrscheinlichkeitsverteilung über eine Variable a
$p(a b)$	Eine Wahrscheinlichkeitsverteilung über a unter der Bedingung b
\mathcal{N}	Eine Normalverteilung
$a \sim p$	Eine Zufallsvariable a mit der Verteilung p
$\mathbb{E}_p[f(x)]$	Der Erwartungswert von $f(x)$ bezüglich $p(x)$
$D_{KL}(p q)$	Die Kullback-Leiber Divergence von p und q

Datensätze

\mathbf{x}	Ein Eingabevektor für das Modell
\mathbf{y} (oder y)	Der Zielwert für die Eingabe \mathbf{x}
\mathbf{x}^*	Ein Eingabevektor für das Modell zur Testzeit
\mathbf{y}^* (oder y^*)	Der/Das Zielwert für die Eingabe \mathbf{x}^*
$\hat{\mathbf{y}}$	Eine Modellausgabe für \mathbf{x}
$\hat{\mathbf{y}}^*$	Eine Modellausgabe für \mathbf{x}^*
$\hat{\mathbf{s}}$	Eine Modellausgabe (Softmaxausgabe) für \mathbf{x}
$\hat{\mathbf{s}}^*$	Eine Modellausgabe (Softmaxausgabe) für \mathbf{x}^*
\mathcal{D}	Ein Datensatz
\mathbf{X}	Eine Matrix aus den Eingabedaten des Datensatzes \mathcal{D}
\mathbf{Y}	Eine Matrix aus den Labels des Datensatzes \mathcal{D}
N	Die Anzahl der Samples im Datensatz \mathcal{D}
\mathbf{x}_n	Die n -te Eingabe aus \mathbf{X} für das Modell
\mathbf{y}_n	Das n -te Label aus \mathbf{Y}
$\hat{\mathbf{y}}_n$	Eine Modellausgabe für \mathbf{x}_n
$\hat{\mathbf{s}}_n$	Eine Modellausgabe (Softmaxausgabe) für \mathbf{x}_n

Aktives Lernen

L	Ein Trainingsdatensatz
L_0	Ein initialer Trainingsdatensatz
U	Ein Datenpool mit ungelabelten Daten
f	Ein neuronales Netzwerk, das mit aktivem Lernen trainiert wird
Q_f	Eine Bewertungsfunktion einer Auswahlstrategie, die f verwendet
b	Die Größe des pro Iteration ausgewählten Samplebatches S

Abkürzungsverzeichnis

Bag	Bagging Ensemble
BNN	Bayessches neuronales Netzwerk
CE	Cross Entropy
CNN	Convolutional Neural Network
CNN c10	CNN für den Datensatz Cifar10
CNN c10 _x	CNN für Cifar10 mit verkleinertem Trainingsdatensatz (x Bilder)
CNN c100	CNN für den Datensatz Cifar100
DA	Data Augmentation Ensemble
DeepAL	Deep Active Learning
DNN	Deep Neural Network
ECE	Expected Calibration Error
Eff	EfficientNet-B3
ELBO	Evidence Lower Bound
KL	Kullback-Leiber
MC	Monte Carlo
MCD	Monte Carlo Dropout
MI	Mutual Information
NUC	Neighborhood Uncertainty Classifier
NUC Tr	Externes Netzwerk vom NUC auf Trainingsdatensatz trainiert
NUC Va	Externes Netzwerk vom NUC auf Validierungsdatensatz trainiert
SE	Softmaxentropie
PE	Prädiktive Entropie

1 Einleitung

Im letzten Jahrzehnt wurden große Fortschritte mit künstlichen neuronalen Netzwerken erreicht. Sie werden aktuell in vielen verschiedenen Forschungsbereichen untersucht, in denen komplexe Systeme modelliert werden müssen, wie beispielsweise bei der Bildanalyse oder Spracherkennung. Obwohl die Netzwerke bereits für die medizinische Bildverarbeitung und für autonome Fahrassistenzsysteme genutzt werden, ist ihre Verwendung in sicherheitskritischen Anwendungen noch begrenzt. Ein Grund hierfür ist unter anderem die fehlende Transparenz neuronaler Netzwerke, was das Vertrauen in die Ausgaben, die sogenannten Prädiktionen, erschwert. Außerdem können neuronale Netzwerke kaum Eingaben der bekannten Domäne von Daten einer unbekanntem Domäne unterscheiden und eine Domänenverschiebung erhöht den Fehler der Netzwerke. Für Anwendungen wie das autonome Fahren sind neuronale Netzwerke daher noch zu unzuverlässig und ungenau. [GTA⁺22]

1.1 Motivation

Um diese Faktoren einzuschränken, ist es wichtig, den Fehler eines Netzwerks mittels zuverlässigen Schätzungen für die Unsicherheit von Ausgaben vorherzusagen. Der amerikanische Professor der Mathematik John Allen Paulos weist dem Begriff der Unsicherheit eine große Bedeutung zu:

„Uncertainty is the only certainty there is, and knowing how to live with insecurity is the only security.“

Das Zitat stammt aus seinem Buch *A Mathematician Plays The Stock Market* [Pau03]. Bezogen auf neuronale Netzwerke wird es kaum ein Netzwerk geben, das immer die richtigen Vorhersagen trifft, weshalb es stets unsichere Ausgaben geben wird. Die Erfassung dieser Unsicherheiten ist eine Möglichkeit mit ihnen zu leben.

Die Uncertainty Estimation in neuronalen Netzwerken befasst sich mit der Frage, wie man die Unsicherheit von neuronalen Netzwerken abschätzen und in den Entscheidungsprozess einbeziehen kann. Ein hohes Uncertainty Estimate kann ein Anzeichen für eine inkorrekte Prädiktion sein und ist ein Indikator dafür, der Prädiktion nicht zu vertrauen und die Daten an Experten und Expertinnen weiterzuleiten.

Neben der Erhöhung des Vertrauens in die Prädiktionen eines neuronalen Netzwerks, kann der Trainingsprozess mittels Uncertainty Estimates optimiert werden. Das aktive Lernen ist ein Prozess, bei dem gezielt Daten für das Netzwerktraining ausgewählt werden, die versprechen, die Genauigkeit des Netzwerks am meisten erhöhen. Damit neuronale Netzwerke eine gewünschte Performance erreichen, werden je nach Komplexität des Problems sehr viele Daten zum Trainieren des Netzwerks benötigt. Der häufig in der

Forschung verwendete Bilddatensatz ImageNet enthält sogar über eine Millionen Bilder [DDS⁺09]. Beim unsicherheitsbasierten aktiven Lernen wird die Unsicherheit von Prädiktionen genutzt, um Netzwerke auf kleinen, ausgewählten Datenmengen zu trainieren und somit den Trainingsprozess zu optimieren. Das ist vor allem in Umgebungen notwendig, in denen die Beschaffung gelabelter Daten teuer oder zeitaufwendig ist. Das Netzwerk entscheidet anhand von Uncertainty Estimates selbst, welche Daten für dessen Training am hilfreichsten und informativsten sind. [Gal16]

Diese Bachelorarbeit ist in Kooperation mit dem *Smart.Production:Lab* der Firma *Volkswagen AG* entstanden. Das *Smart.Production:Lab* ist eine Abteilung des Volkswagenwerks in Wolfsburg. Ein großer Forschungsbereich des Labs ist die Computer Vision, auch maschinelles Sehen genannt. Computer Vision ist ein Teilgebiet der Computervisualistik, der sich mit der automatischen Verarbeitung, Analyse und Interpretation von Bildern und anderen visuellen Eingaben beschäftigt [Pri15]. Hierfür werden im *Smart.Production:Lab* neuronale Netzwerke eingesetzt. Um den Labelaufwand für das Training dieser Netzwerke zu reduzieren und somit der Volkswagen AG Kosten zu ersparen, wird überlegt, in Zukunft aktives Lernen in den Trainingsprozess einzubeziehen.

1.2 Zielsetzung

Viele Forscher und Forscherinnen haben bereits verschiedene Methoden und Techniken entwickelt, um die Unsicherheit neuronaler Netzwerke abzuschätzen. Für das aktive Lernen im *Smart.Production:Lab* soll möglichst die beste Methode umgesetzt werden. Jedoch ist aufgrund vieler Kriterien, anhand welcher die Methoden verglichen werden können, nicht direkt klar, welche Methode am besten geeignet ist. Es gibt kein standardisiertes Protokoll mit Tests, die von Methoden zur Uncertainty Estimation ausgeführt werden sollten, um sie vergleichen zu können. Viele Paper führen zwar prinzipiell gleiche Tests durch, jedoch variieren die Details der Experimente von Paper zu Paper. [GTA⁺22]

Ziel dieser Arbeit ist es, häufig verwendete Tests mit ausgewählten Methoden zur Uncertainty Estimation durchzuführen, um die Methoden mit exakt den gleichen Experimenten angemessen vergleichen zu können. Unabhängig davon wird der Prozess des unsicherheitsbasierten aktiven Lernens für ein neuronales Netzwerk simuliert und untersucht, mittels welcher Methode der schnellste Performanewachstum zu vermerken ist. Es soll herausgefunden werden, ob Methoden, die in den Tests besonders gute Ergebnisse liefern, also qualitativ hochwertige Uncertainty Estimates vorhersagen, ebenfalls am besten für das aktive Lernen geeignet sind. Somit soll geklärt werden, ob die häufig verwendeten Test zur Bewertung von Uncertainty Estimates repräsentativ für den konkreten Anwendungsbereich, das aktive Lernen, sind.

Spezielle Anwendungsbereiche neuronaler Netzwerke im *Smart.Production:Lab* sind unter anderem die Klassifizierung und semantische Segmentierung. Bei der Klassifizierung wird jeder Eingabe eine Kategorie zugeordnet [GBC17]. Die Problemstellung der semantischen Segmentierung ist die Einteilung eines digitalen Bildes in inhaltlich zusammenhängende Bereiche, indem jedem Pixel eine Klasse zugeordnet wird [CVC23]. Da die semantische Segmentierung auf der einfachen Klassifizierung aufbaut, untersucht diese Arbeit Methoden zur Uncertainty Estimation für Klassifizierungsprobleme, um abschätzen zu können, ob die Verwendung des aktiven Lernens bei Volkswagen Potential hat. Die Frage inwiefern

die Ergebnisse auf die semantische Segmentierung übertragbar sind, kann Gegenstand zukünftiger Untersuchungen sein.

1.3 Gliederung der Arbeit

Diese Arbeit ist in insgesamt fünf Kapitel gegliedert. Im ersten (diesem) Kapitel werden Motivation und Zielsetzung der Arbeit beschrieben. Zudem werden verwandte Arbeiten vorgestellt. Das zweite Kapitel liefert die zum Verständnis der Arbeit notwendigen Grundlagen. Hier werden außerdem die Methoden zur Uncertainty Estimation vorgestellt. Zu den Grundlagen gehören fundamentale Kenntnisse künstlicher neuronaler Netzwerke und eine Einführung in den Begriff der prädiktiven Unsicherheit. Anschließend wird ein Überblick der Methoden des State of the Art gegeben, bevor die grundlegende wahrscheinlichkeitstheoretische Modellierung der prädiktiven Unsicherheit durch bayessche neuronale Netzwerke genauer erläutert wird. Es folgen detaillierte Beschreibungen konkreter Methoden. Abgeschlossen wird das zweite Kapitel mit der Beschreibung von Metriken, die auf Basis solcher Methoden Uncertainty Estimates für Klassifizierungsprobleme ermitteln. Im dritten Kapitel werden die Methoden anhand häufig angewandter Tests evaluiert. Zunächst erfolgt die Beschreibung der neuronalen Netzwerke und Datensätze, auf welche die Methoden angewandt werden. Die anschließenden Tests untersuchen die Qualität der Uncertainty Estimates anhand verschiedener Aspekte. Das daran anschließende vierte Kapitel betrachtet die Verwendung der Methoden für das aktive Lernen. Es folgt ein Überblick des Trainings mit dem unsicherheitsbasierten aktiven Lernen, bevor der Prozess für ein konkretes Klassifizierungsnetzwerk mit den betrachteten Methoden durchgeführt wird. Anschließend erfolgen Untersuchungen zur weiteren Performanceoptimierung. Im letzten Kapitel werden die Erkenntnisse dieser Arbeit zusammengefasst und ein Fazit gezogen. Es folgt ein Einblick in die Uncertainty Estimation bei semantischer Segmentierung und ein Überblick zu aktuellen Grenzen und Problematiken der Uncertainty Estimation. Dabei wird ein Ausblick gegeben und zum Schluss wird die potenzielle Verwendung des aktiven Lernens bei Volkswagen diskutiert.

1.4 Verwandte Arbeiten

Es gibt weitere Arbeiten, deren Schwerpunkt auf dem Vergleich verschiedener Methoden zur Uncertainty Estimation liegt. Sie führen zur Evaluierung meist nur einzelne Tests durch und betrachten somit wenig Kriterien. Außerdem verwendet der Großteil der Arbeiten ausschließlich simple Datensätze wie MNIST und Cifar10. Wiederum andere Arbeiten untersuchen die Verwendung verschiedener Methoden für das aktive Lernen unabhängig von deren Performance bei konkreten Evaluierungstests.

M. Henne et al. [HSRW20] evaluieren ausgewählte Methoden zu Uncertainty Methoden ebenfalls für Klassifizierungsprobleme. Sie betrachten die Kalibrierung und wie gut korrekte und inkorrekte Prädiktionen anhand der Uncertainty Estimates erkannt werden können. Jedoch untersuchen sie weder die Performance bezüglich der Erkennung von Daten unbekannter Domänen noch den Rechenaufwand und die Laufzeit der Methoden.

A. Hein et al. [HRG⁺22] vergleichen die Methoden für das aktive Lernen. Hierfür simulieren sie ebenfalls den Prozess des aktiven Lernens für Klassifizierungsnetzwerke und untersuchen, welche Methode zum schnellsten Performancewachstum der Netzwerke führt. Sie wählen dafür stets die 10 unsichersten Daten aus einer Menge ungelabelter Daten aus, die anschließend gelabelt und zum Trainingsdatensatz hinzugefügt werden, woraufhin das Netzwerk erneut trainiert wird. Nun beginnt die Auswahl erneut. Die Wahl von nur 10 Daten erscheint ineffizient, da das Labeln von 10 Klassifizierungsdaten in weniger als einer Minute erfolgen und das anschließende Netzwerktraining deutlich länger dauern kann. Folglich könnten in jeder Iteration mehr Daten für das Labeling hinzugefügt werden, um schneller eine gewünschte Netzwerkperformance zu erreichen. A. Hein et al. evaluieren die Methoden außerdem kaum für die eigentliche Uncertainty Estimation. Herauszufinden, ob die Performance beim aktiven Lernen mit den Ergebnissen von Evaluierungstests der Uncertainty Estimation zusammenhängt, ist wichtig, da die Bewertung der Methoden für das aktive Lernen mit einem hohem Rechenaufwand verbunden ist.

P. S. Wespe [Wes23] hat bereits exemplarisch untersucht wie eine Methode zur Uncertainty Estimation im Smart.Production:Lab integriert werden kann. Die untersuchte Methode ist das Monte Carlo Dropout. Das entwickelte Framework setzt die Grundlage für die Verwendung beliebiger Methoden zum aktiven Lernen bei Volkswagen.

2 Grundlagen

Dieses Kapitel umfasst das Hintergrundwissen, welches für den Vergleich verschiedener Uncertainty Estimation Methoden wichtig ist. Abschnitt 2.1 gibt zunächst eine Einführung in die Funktionsweise neuronaler Netzwerke. Anschließend wird in Abschnitt 2.2 die Problematik der prädiktiven Unsicherheit definiert und erläutert. Der aktuelle Stand der Forschung für die Uncertainty Estimation wird in Abschnitt 2.3 vorgestellt. Hier werden ausgewählte Techniken erläutert, die für den Vergleich interessant sind. Der Fokus liegt auf häufig für die Klassifizierung verwendeten Methoden und auf Ansätzen, die in ihrem Aufbau und ihrer Funktionsweise stark variieren. Wie mittels der Methoden speziell für Klassifizierungsprobleme die prädiktive Unsicherheit abgeschätzt werden kann, wird in Abschnitt 2.4 erklärt.

2.1 Künstliche neuronale Netzwerke

Durch künstliche neuronale Netzwerke lassen sich unterschiedliche Datenstrukturen wie Bilder, Texte, Tabellen oder Zeitreihen interpretieren und Informationen oder Muster extrahieren, um diese auf unbekannte Daten anzuwenden. Die Netzwerke berechnen so Vorhersagen für die unbekanntenen Daten.

Im Folgenden werden Konzepte und Eigenschaften neuronaler Netzwerke, die für die Arbeit wichtig sind, geklärt. Es wird auf die grundlegenden Netzwerke, die Deep Feedforward Networks, sowie auf die fortgeschrittenen Convolutional Neural Networks (CNNs) eingegangen. Beide Modellarchitekturen zählen zu den Deep Neural Networks (DNNs). Außerdem wird beschrieben, wie die Modelle für Klassifizierungsprobleme angewandt werden. Die Erklärungen sind am Buch *Deep Learning* [GBC17] orientiert.

2.1.1 Aufbau von Deep Feedforward Networks

Deep Feedforward Networks erzielen die Approximation einer Funktion f^* . Diese ordnet jeder Eingabe \mathbf{x} einen Zielwert $\mathbf{y} = f^*(\mathbf{x})$ zu. Ein Feedforward Network definiert die Abbildung $f_\omega : \mathcal{X} \rightarrow \mathcal{Y}$ mit $\hat{\mathbf{y}} = f_\omega(\mathbf{x})$. Das Netzwerk erzielt die Abbildung einer Eingabe \mathbf{x} der Domäne \mathcal{X} auf den zugehörigen Zielwert \mathbf{y} aus der Domäne \mathcal{Y} . Die Netzwerkausgabe $\hat{\mathbf{y}}$ ist durch das Zirkumflexzeichen gekennzeichnet. Das Netzwerk erlernt Werte der Parameter ω , die die Funktion f^* möglichst gut approximieren.

Solche Modelle werden *Feedforward* genannt, weil die Informationen einer Eingabe durch mehrere sequentiell ausgeführte Transformationen durch das Netzwerk fließen und anschließend in eine Ausgabe umgewandelt werden. DNNs bestehen aus einer Komposition mehrerer verschiedener Funktionen, zum Beispiel $f(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$. Dies sind die sogenannten Schichten des neuronalen Netzwerks. Die erste Schicht ist die Eingabe- und

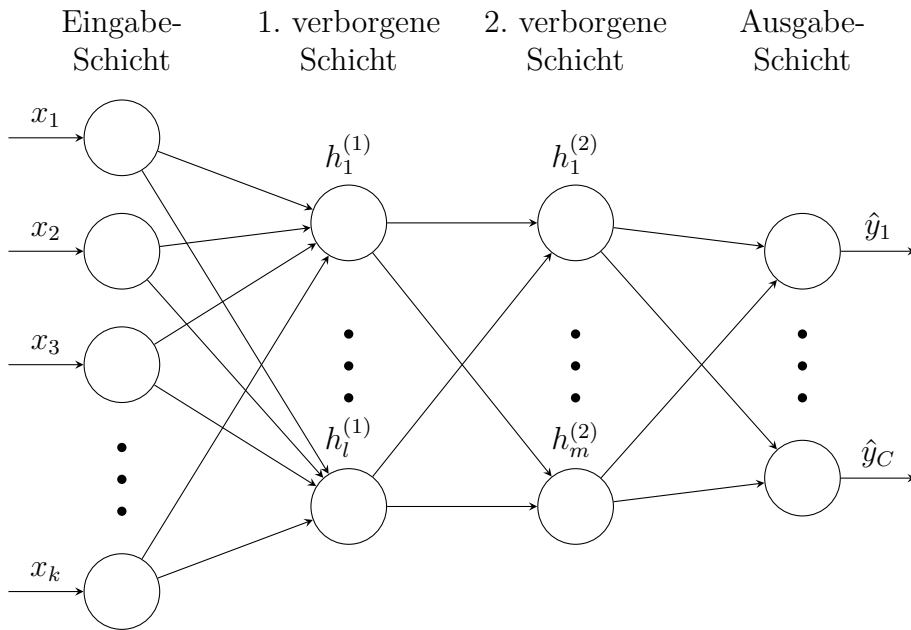


Abbildung 2.1: Skizzierte Architektur eines Deep Feedforward Networks mit zwei verborgenen Schichten. Die Pfeile repräsentieren die Multiplikation der Ausgabe eines Neurons mit einem Gewicht. Alle Eingaben eines Neurons werden summiert, ein Bias dazu addiert und das Resultat durch eine nicht-lineare Aktivierungsfunktion $g^{(i)}$ transformiert.

die letzte die Ausgabeschicht. Alle Schichten dazwischen sind verborgene Schichten, die im Folgenden mittels \mathbf{h} gekennzeichnet werden. Die Netzwerke werden als *deep* bezeichnet, wenn sie über mindestens zwei Schichten verfügen, was in der Praxis der Fall ist. Jede Schicht besteht aus mehreren Neuronen, die parallel fungieren. Eingaben eines Neurons sind die faktorisierten Ausgaben der Neuronen aus der Schicht davor. Die Faktoren werden als Gewichte bezeichnet und die Gewichte einer gesamten Schicht sind in einer Gewichtsmatrix \mathbf{W} gespeichert. In Abbildung 2.1 ist die allgemeine Netzwerkstruktur eines Deep Feedforward Networks skizziert. Die genaue Transformation der Eingabe \mathbf{x} bis zur Ausgabe $\hat{\mathbf{y}}$ kann folgendermaßen formal definiert werden:

$$\mathbf{h}^{(1)} = g^{(1)}(\mathbf{W}^{(1)T} \cdot \mathbf{x} + \mathbf{b}^{(1)}), \quad (2.1)$$

$$\mathbf{h}^{(i)} = g^{(i)}(\mathbf{W}^{(i)T} \cdot \mathbf{h}^{(i-1)} + \mathbf{b}^{(i)}), \quad (2.2)$$

$$\hat{\mathbf{y}} = g^{(n)}(\mathbf{W}^{(n)T} \cdot \mathbf{h}^{(n-1)} + \mathbf{b}^{(n)}). \quad (2.3)$$

Die Multiplikation mit der Gewichtsmatrix $\mathbf{W}^{(i)}$ und die anschließende Addition eines sogenannten Biasvektors $\mathbf{b}^{(i)}$ ergeben eine lineare Transformation. Um beliebige Funktionen f^* approximieren zu können, sind jedoch zusätzliche nicht-lineare Transformationen $g^{(i)}$, die Aktivierungsfunktionen, notwendig. Für verborgene Schichten wird häufig die Aktivierungsfunktion $g(z) = \max\{0, z\}$ verwendet. Ein Neuron mit dieser Aktivierungsfunktion wird Rectified Linear Unit (ReLU) genannt. Die Funktion ist einer linearen sehr ähnlich, weshalb die Neuronen leicht zu optimieren sind. Die Aktivierungsfunktion der Ausgabeschicht ist speziell an den Anwendungsbereich des Netzwerks angepasst. Für die Klassifizierung werden meistens Softmaxfunktionen verwendet, um zwischen C Klassen zu unterscheiden. Die Ausgabeschicht enthält in diesem Fall C Neuronen, eins für jede

Klasse. Für eine Klasse j wird die Softmaxausgabe, auch Softmaxscore genannt, berechnet als

$$\text{softmax}(\mathbf{z})_j = \frac{\exp(z_j)}{\sum_{c=1}^C \exp(z_c)}, \quad (2.4)$$

wobei \mathbf{z} die Ausgabe der linearen Transformation $\mathbf{W}^{(n)T} \cdot \mathbf{h}^{(n-1)} + \mathbf{b}^{(n)}$ der letzten Schicht ist. Sie liegt stets zwischen 0 und 1 und die Addition aller Softmaxscores der letzten Schicht ergibt 1. Die Klasse, deren Softmaxscore am höchsten ist, wird vom Netzwerk vorhergesagt. Sie ist die Prädiktion \hat{y} (Vorhersage) für die Eingabe \mathbf{x} . Hier ist der Vektor $\hat{\mathbf{y}}$ nur eindimensional. Unterscheidet ein Klassifizierungsproblem lediglich zwischen zwei Klassen, kann alternativ die Sigmoidfunktion $g(z) = \frac{1}{1+e^{-z}}$ verwendet werden. Ihr Wertebereich ist ebenfalls $(0, 1)$ und hier gibt es nur ein Neuron in der Ausgabeschicht. Ist die Sigmoidausgabe größer als 0.5, wird sich für die eine Klasse, ansonsten für die andere Klasse entschieden.

2.1.2 Training künstlicher neuronaler Netzwerke

Beim Training eines Deep Neural Networks (DNNs) wird die Anpassung von $f(\mathbf{x})$ an $f^*(\mathbf{x})$ erzielt. Die Trainingsdaten enthalten approximierete Beispiele von $f^*(\mathbf{x})$. Jedem \mathbf{x} wird ein passendes Label $\mathbf{y} \approx f^*(\mathbf{x})$ zugeordnet¹. Bei Klassifizierungsproblemen ist das Label y eines der C Klassen $y \in \{1, 2, 3, \dots, C\}$ oder die Klasse wird als sogenannter One-Hot Vektor mit C Einträgen repräsentiert. In diesem Fall ist $\mathbf{y} = (0, \dots, 0, 1, 0, \dots, 0)$ mit dem Wert 1 am Index der Klasse. Die beiden Notationen werden nun am Beispiel des MNIST-Datensatzes, der aus Bildern handgeschriebener Ziffern besteht, verdeutlicht. Alle Bilder haben 28×28 Pixel, sodass die Eingaben \mathbf{x} des Netzwerks Vektoren mit 28×28 Einträgen aus dem Bereich der reellen Zahlen \mathbb{R} sind. Folglich haben sie 784 Werte. Ein One-Hot-kodiertes Label \mathbf{y} hat 10 ganzzahlige Einträge, da es 10 Ziffern und somit 10 Klassen gibt.

Die letzte Schicht des neuronalen Netzwerks soll für jeden Datenpunkt \mathbf{x} Ausgaben $\hat{\mathbf{y}}$ erzeugen, die nah bei \mathbf{y} liegen. Das Verhalten der anderen Schichten ist nicht durch die Trainingsdaten spezifiziert. Der Trainingsalgorithmus muss entscheiden, wie diese Schichten verwendet werden. Hierfür werden die Parameter $\boldsymbol{\omega}$ des Netzwerks $f_{\boldsymbol{\omega}}(\mathbf{x})$ verändert. Zu Beginn des Trainings werden alle Gewichte zufällig und die Bias-Vektoren mit 0 initialisiert. Die Trainingsdaten seien im gelabelten Datensatz $\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\}$ zusammengefasst. Er enthält N Eingabedaten $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$ mit den zugehörigen Labels $\mathbf{Y} = \{\mathbf{y}_n\}_{n=1}^N$. Daten, die in der Testphase verwendet werden, sind mit einem Stern gekennzeichnet. Sei die Testeingabe \mathbf{x}^* gegeben, wird in dieser Phase erwartet, dass die Netzwerkausgabe $\hat{\mathbf{y}}^*$ mit dem für das Netzwerk unbekanntem Label \mathbf{y}^* übereinstimmt. Beziehungsweise soll bei der One-Hot-Kodierung für die Klassifizierung gelten, dass die vorhergesagte Klasse von $\hat{\mathbf{y}}^*$ der von \mathbf{y}^* entspricht.

Das Training künstlicher neuronaler Netze kann aus einer probabilistischen Perspektive beschrieben werden. Die nachfolgenden Erklärungen aus dieser Perspektive sind an D. Feng [Fen21] angelehnt. Ziel des Trainings ist das Finden der Parameter $\boldsymbol{\omega}$ des Netzwerks $f_{\boldsymbol{\omega}}(\mathbf{x})$, die am wahrscheinlichsten die Labels generieren. Um dies herauszufinden, muss

¹Mit dem Begriff *Training* ist in dieser Arbeit stets das überwachte Training (*supervised Learning*) gemeint, bei dem für jede Eingabe \mathbf{x} gelabelte Zielwerte \mathbf{y} zur Verfügung stehen. Dies ist beim *Unsupervised Learning* nicht der Fall.

eine Wahrscheinlichkeitsverteilung $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega})$ gefunden werden, die die Wahrscheinlichkeit beschreibt, dass aus einer Eingabe \mathbf{x} mit den Parametern $\boldsymbol{\omega}$ das Label \mathbf{y} generiert wird. Für Klassifizierungsprobleme sind es die Softmaxscores:

$$p(y = c|\mathbf{x}, \boldsymbol{\omega}) = \text{softmax}(\mathbf{z})_j = \frac{\exp(z_j)}{\sum_{c=1}^C \exp(z_c)}. \quad (2.5)$$

Die vorhergesagte Klasse ist die mit dem größten Softmaxscore \hat{s}_c :

$$\hat{y} = \underset{c \in \{1, \dots, C\}}{\text{argmax}}(\hat{s}_c). \quad (2.6)$$

Die Netzwerkparameter $\boldsymbol{\omega}$ können mittels des Maximum Likelihood (ML) Schätzers bestimmt werden. Er maximiert die Wahrscheinlichkeit $p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega})$ der Trainingsdaten. Angenommen die einzelnen Eingabedaten, die Samples \mathbf{x} , sind voneinander unabhängig, so gilt:

$$p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega}) = \prod_{n=1}^N p(\mathbf{y}_n|\mathbf{x}_n, \boldsymbol{\omega}). \quad (2.7)$$

In der Praxis wird im Verlauf des Trainings die negative, logarithmische Likelihood (Log Likelihood) minimiert. Die Funktion, die die Abweichung der aktuellen Parameter $\boldsymbol{\omega}$ von den unbekannt optimalen Parametern misst, wird Lossfunktion genannt. Sie ist definiert als:

$$L(\boldsymbol{\omega}) = -\log(p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega})). \quad (2.8)$$

Für Klassifizierungsprobleme resultiert eine Maximum Likelihood Schätzung in der Cross Entropy (CE) Lossfunktion. Gibt es nur zwei Klassen, so sei \hat{s} der Score für die korrekte Klasse. Der CE Loss berechnet sich als:

$$L_{CE}(\boldsymbol{\omega}) = -\sum_{n=1}^N (y_n \log(\hat{s}_n) + (1 - y_n) \log(1 - \hat{s}_n)). \quad (2.9)$$

Bei mehreren Klassen wird der Loss für jede Klasse einzeln berechnet und anschließend werden alle addiert.

Da Trainingsdatensätze in der Praxis sehr groß sein können und die Ermittlung des gesamten Loss somit zu aufwendig ist, teilt man den Datensatz in mehrere kleine Teilmengen, die Minibatches, ein. Sequentiell werden für jeden Minibatch einzeln zunächst der Loss ermittelt und anschließend die Parameter $\boldsymbol{\omega}$ dem Loss entsprechend upgedatet. Wurden alle Minibatches verwendet, so ist eine sogenannte Trainingsepoche vorbei und der Prozess beginnt erneut.

Für die Updates wird der Back-Propagation Algorithmus verwendet, der Optimierungsmethoden wie stochastische Gradientenverfahren nutzt. Grundlage dieser Verfahren ist der Gradientenabstieg. Beim Gradientenabstieg werden die Parameter $\boldsymbol{\omega}$ in Richtung des negativen Gradienten der Lossfunktion $\nabla L(\boldsymbol{\omega})$ verändert, da dieser zu einem lokalen Minimum führt. Wie weit sich in diese Richtung bewegt wird, bestimmt die Lernrate η . Um sich schneller einem Optimum anzunähern und nicht in ein verhältnismäßig schlechtes lokales Optimum zu konvergieren, wird zu Beginn des Trainings eine hohe Lernrate verwendet, welche im Verlauf des Trainings verringert wird. Abbildung 2.2 illustriert den

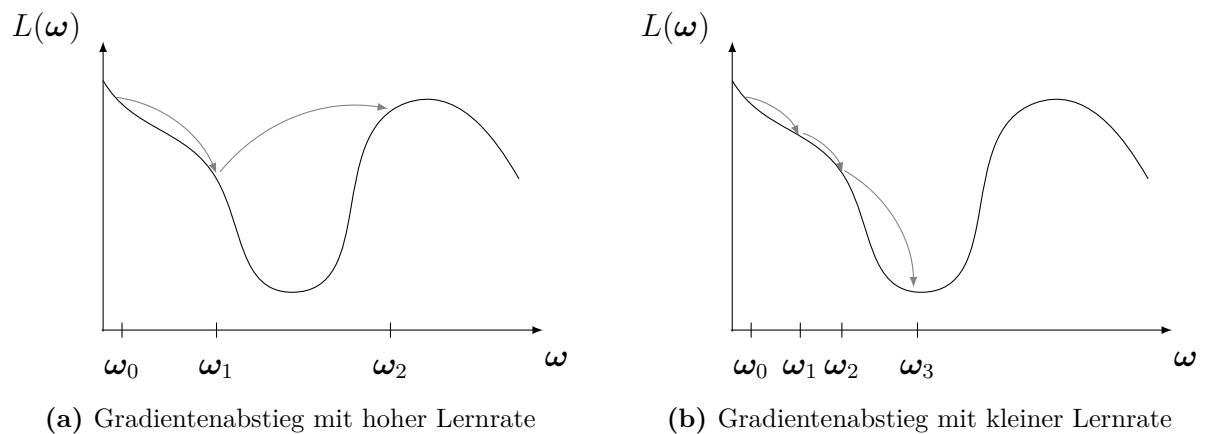


Abbildung 2.2: Darstellung der ersten Schritte des Gradientenabstiegs in einem eindimensionalen Parameterraum. Der Gradient $\nabla L(\omega)$ ist in diesem Fall die einfache Ableitung von $L(\omega)$ nach ω . Die Veränderung des Parameters ist einerseits mit einer hohen Lernrate **(a)** und andererseits mit einer kleineren Lernrate **(b)** beispielhaft skizziert.

Gradientenabstieg anhand von zwei Beispielen für einen stark vereinfachten eindimensionalen Parameterraum².

Die Overfitting-Problematik beim Training von DNNs

Ziel beim Training eines DNNs ist, dass das Netzwerk nicht nur auf dem Trainingsdatensatz gute Ergebnisse erzielt, sondern auch für neue, unbekannte Eingaben \mathbf{x}^* zur Testzeit häufig die richtigen Prädiktionen $\hat{\mathbf{y}}^*$ berechnet. Die Fähigkeit eines Netzwerks, eine hohe Performance für neue Daten zu erreichen, nennt sich Generalisierung. Mit der Lossfunktion kann die Abweichung der Netzwerkausgaben von den Labels gemessen werden. Für den Trainingsdatensatz nennt man die Abweichung Trainingsfehler. Der erwartete Fehler des Netzwerks bei neuen Eingaben ist der Generalisierungsfehler, auch als Testfehler bezeichnet. Um die Modellperformance, beispielsweise die Genauigkeit (Accuracy), für neue Daten zu messen, wird ein Testdatensatz $\mathcal{D}^* = \{\mathbf{X}^*, \mathbf{Y}^*\}$ verwendet. Er enthält M für das DNN unbekannte Eingabedaten $\mathbf{X}^* = \{\mathbf{x}_m^*\}_{m=1}^M$ und zugehörige Labels $\mathbf{Y}^* = \{\mathbf{y}_m^*\}_{m=1}^M$. Ziel des Trainings neuronaler Netzwerke ist es, nicht nur einen geringen Trainingsfehler zu erhalten, sondern auch die Differenz von Trainings- und Testfehler gering zu halten.

Deep Neural Networks haben eine sehr hohe repräsentative Kapazität, was bedeutet, dass sie hinreichend viele Parameter haben, um sehr komplexe Funktionen f zu modellieren. Zu Beginn des Trainings fällt sowohl der Trainings- als auch der Testfehler. Der Testfehler sinkt irgendwann langsamer als der Trainingsfehler. Eine grobe Erklärung hierfür ist, dass sich einzelne Parameter des DNNs immer mehr auf das Vorhandensein sehr individueller Eigenschaften der Trainingsdaten fokussieren, um den Trainingsfehler weiter zu verringern. Für die begrenzte Anzahl an Trainingsdaten gibt es oft mehr Parameter als

²Bemerke, dass der eigentliche Back-Propagation Algorithmus deutlich komplexer ist. Zur Ermittlung des Gradienten muss der individuelle Einfluss jedes Parameters an der Ausgabe bestimmt werden, damit jeder Parameter korrekt upgedatet werden kann.

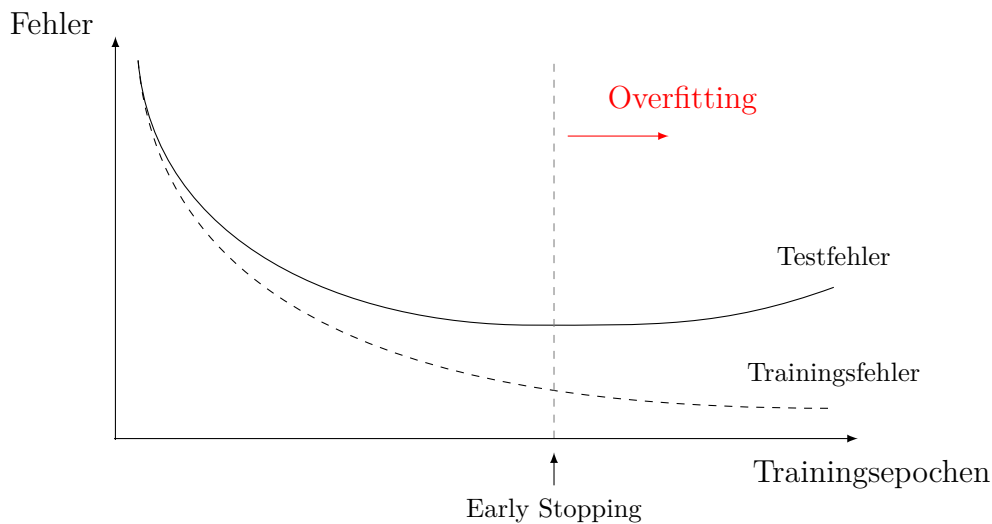


Abbildung 2.3: Skizzierte Entwicklung von Trainings- und Testfehler im Verlauf des Trainings. Die favorisierten Modellparameter sind die, für die der Testfehler am geringsten ist. An diesem Punkt sollte das Training beendet werden, um Overfitting zu vermeiden.

für eine gute Generalisierung ausreichend wäre. Die unerwünschte Situation, in der der Trainingsfehler langsamer fällt als die Differenz zwischen Test- und Trainingsfehler steigt, wird Overfitting genannt. Ab diesem Punkt steigt der Testfehler wieder. Abbildung 2.3 visualisiert die Entwicklung von Trainings- und Testfehler im Verlauf des Trainingsprozesses.

An dem Punkt, an dem der Testfehler am geringsten ist, sollte das Training im Idealfall beendet werden. Jedoch ist es wichtig, dass die Testdaten keinesfalls das Modelltraining beeinflussen oder für Entscheidungen über das Modell verwendet werden dürfen. Nur so entspricht der Fehler für den Testdatensatz dem erwarteten Fehler für neue Daten. Daher werden alle Trainingsdaten vor Beginn des Trainings in zwei disjunkte Mengen aufgeteilt. Die eine bildet den vom Trainingsalgorithmus verwendeten Trainingsdatensatz zum Lernen der Parameter und die andere bildet einen Validierungsdatensatz, der zur Bestimmung des Generalisierungsfehlers während des Trainings genutzt werden kann. Anhand der Entwicklung des Fehlers auf den Validierungsdaten wird entschieden, wann das Training zu beenden ist. Diese Vorgehensweise nennt sich Early Stopping.

Mit dem Fehler für die Validierungsdaten kann außerdem bestimmt werden, wann die Lernrate des Trainingsalgorithmus verringert werden soll. Hat sich der Fehler einige Epochen hinweg nicht mehr verbessert, ist es hilfreich, die Lernrate zu reduzieren, damit sich einem lokalen Optimum der Lossfunktion in kleineren Schritten genähert und nicht wieder davon entfernt wird, wie es in Abbildung 2.2a der Fall ist. Verringert sich der Fehler weiterhin nicht, sollte das Training beendet werden.

Es gibt noch viele weitere Strategien, die die Reduzierung des Testfehlers erzielen, oft mit der Konsequenz eines erhöhten Trainingsfehlers. Diese Strategien werden Regularisierungstechniken genannt. Dazu gehören Beschränkungen des Wertebereichs der Parameter oder das Hinzufügen bestimmter Terme zur Lossfunktion.

2.1.3 Grundlagen Convolutional Neural Networks (CNNs)

Convolutional Neural Network (CNNs) sind eine spezielle Architektur unter den künstlichen neuronalen Netzwerken. Sie verarbeiten Daten, die eine regelmäßige Struktur haben. Ein prominentes Beispiel hierfür sind unter anderem Bilddaten, da Bilder aus einem zweidimensionalen Pixelgitter bestehen, weshalb CNNs häufig zur Mustererkennung in Bildern verwendet werden [ON15]. CNNs verwenden eine spezielle lineare mathematische Operation, die *Convolution* genannt wird. Ein Netzwerk ist ein CNN, wenn es Convolution an Stelle der Matrixmultiplikation in mindestens einer Schicht verwendet. Eine solche Schicht wird als Convolutional-Schicht bezeichnet und eine mit der gewöhnlichen Matrixmultiplikation als vollständig verbundene Schicht.

Ein Vorteil der Convolutional-Schicht ist, dass die zweidimensionale Anordnung der Pixel als Eingabe erhalten bleiben kann. Die Convolution-Operation fokussiert sich auf lokal beieinander liegende Pixel und erkennt dadurch leicht bestimmte Muster, weshalb CNNs meistens eine höhere Performance aufweisen als Deep Feedforward Networks. Sie sind zusätzlich effizienter, da nicht wie in einer vollständig verbundenen Schicht für jede Ausgabe eines Neurons jeder Wert der Eingabe, in diesem Beispiel jeder Pixel, betrachtet wird. Somit werden weniger Parameter benötigt.

Für zweidimensionale Daten I kann die Convolution-Operation folgendermaßen definiert werden:

$$S(i, j) = (K * I)(i, j) = \sum_m \sum_n I(i + m, j + n)K(m, n). \quad (2.10)$$

Die Funktion K ist der *Kernel*. Im Kernel sind alle Parameter der Convolutional-Schicht enthalten. Er ist meistens deutlich kleiner als I . Über die Größe des Kernels wird mittels m und n iteriert. Die Ausgabe S ist wie die Eingabe I und der Kernel K zweidimensional und wird *Feature Map* genannt. Abbildung 2.4 demonstriert die Berechnung an einem Beispiel.

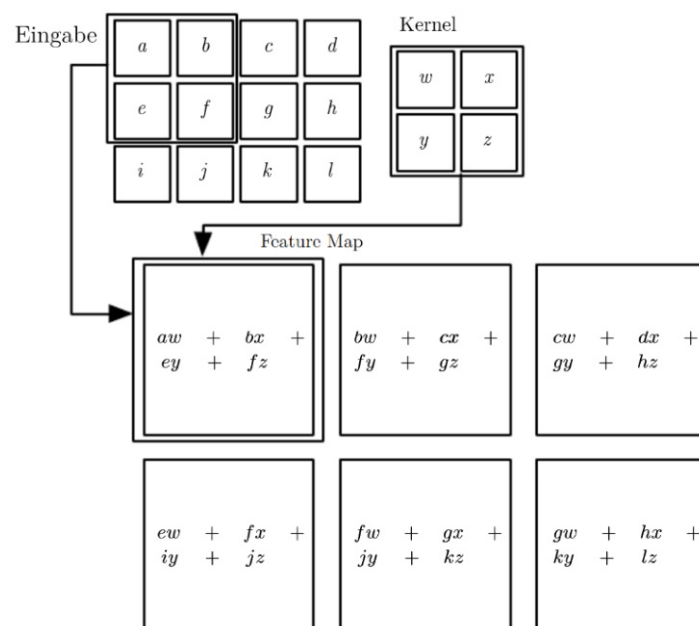


Abbildung 2.4: Beispielhafte Ausführung der Convolution-Operation mit einem Kernel der Größe 2×2 (Grafik von [GBC17]).

Convolutional-Schichten können mehr als einen Kernel enthalten, für die die Convolution-Operation separat ausgeführt wird. Somit erzeugen die Convolutional-Schichten mehrere Feature Maps. Anschließend wird wie bei den vollständig verbundenen Schichten eine nicht-lineare Aktivierungsfunktion g elementweise angewandt. Oftmals folgt nun die Anwendung einer Pooling-Funktion. Eine solche Funktion ersetzt eine Ausgabe durch eine statistische Zusammenfassung der Ausgaben in unmittelbarer Nähe, beispielsweise durch das Maximum oder das arithmetische Mittel. Dies verkleinert die Ausgabe und macht sie invariant für Translationen der Eingabe. Wenn es für ein Problem wichtiger ist, zu wissen ob ein bestimmtes Merkmal (ein Feature) in der Eingabe existiert als zu wissen, an welcher exakten Position es sich befindet, sind Pooling-Funktionen besonders nützlich.

2.2 Prädiktive Unsicherheit in DNNs

Die Uncertainty Estimation erzielt die Modellierung prädiktiver Unsicherheiten eines Modells zur Testzeit. In diesem Kapitel soll Verständnis für den Begriff der prädiktiven Unsicherheit geschaffen werden. Dabei werden Ursachen erläutert und häufig verwendete Unterteilungen der prädiktiven Unsicherheit vorgestellt. Die Erklärungen sind an J. Gawlikowski et al. [GTA⁺22] angelehnt.

2.2.1 Definition der prädiktiven Unsicherheit eines DNNs

Die Unsicherheit einer Modellvorhersage, der Prädiktion $\hat{\mathbf{y}}^*$, für eine bestimmte Testeingabe \mathbf{x}^* wird als prädiktive Unsicherheit bezeichnet. Der Stern an den Vektoren deutet daraufhin, dass es sich bei \mathbf{x}^* um eine Eingabe handelt, die nicht für das Training des Deep Neural Networks (DNNs) verwendet wurde. Was genau mit dem Begriff der prädiktiven Unsicherheit gemeint ist, wird im Folgenden geklärt.

Die Wahrscheinlichkeit einer Ausgabe \mathbf{y}^* der Funktion f^* für eine bestimmte Testeingabe \mathbf{x}^* sei $p(\mathbf{y}^*|\mathbf{x}^*)$. Die optimale Ausgabe sei gegeben als:

$$\mathbf{y}^* = \underset{\mathbf{y}}{\operatorname{argmax}} p(\mathbf{y}|\mathbf{x}^*). \quad (2.11)$$

Falls $p(\mathbf{y}^*|\mathbf{x}^*)$ diskret ist, ist \mathbf{y}^* der Datenpunkt mit der höchsten Wahrscheinlichkeit bzw. im stetigen Fall der Punkt mit der höchsten Wahrscheinlichkeitsdichte. Allerdings ist $p(\mathbf{y}^*|\mathbf{x}^*)$ nicht bekannt und somit auch die optimale Ausgabe nicht. Die Verteilung $p(\mathbf{y}^*|\mathbf{x}^*)$ kann jedoch approximiert werden. Hierfür wird ein Trainingsdatensatz als Stichprobe $\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\}$ aus der realen Welt entnommen, der N Samples $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$ mit zugehörigen Labels $\mathbf{Y} = \{\mathbf{y}_n\}_{n=1}^N$ enthält. \mathcal{D} ist genau genommen nur eine vereinfachte Repräsentation einer Stichprobe der realen Welt. Beispielsweise beschränkt die Anzahl der Pixel einer Bildaufnahme die Genauigkeit der dargestellten Situation. Die Menge an Daten des von Modellen genutzten, vereinfachten Wertebereiches wird im Folgenden mit D bezeichnet. Der Datensatz \mathcal{D} enthält eine Teilmenge der Daten dieses Wertebereiches. Die Wahrscheinlichkeitsverteilung $p(\mathbf{y}^*|\mathbf{x}^*)$ kann nun folgendermaßen approximiert werden:

$$p(\mathbf{y}^*|\mathbf{x}^*) \approx \int_D p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D}) d\mathcal{D}. \quad (2.12)$$

Die Verteilung $p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D})$, auch $p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y})$, wird prädiktive Wahrscheinlichkeitsverteilung genannt. Ein Schätzer für die bevorzugte Ausgabe ist

$$\mathbf{y}^* \approx \underset{\mathbf{y}}{\operatorname{argmax}} p(\mathbf{y}|\mathbf{x}^*, \mathcal{D}). \quad (2.13)$$

Gleichung 2.12 kann nur basierend auf D abgeschätzt werden. Für diese Abschätzung eignen sich DNNs besonders. Ein DNN f ist theoretisch in der Lage eine komplexe Funktion f^* gut zu approximieren. Jedoch ist durch die Approximation und die begrenzte Anzahl an Daten in \mathcal{D} nicht sichergestellt, dass die Prädiktion $\hat{\mathbf{y}}^*$ eines DNNs mit \mathbf{y}^* übereinstimmt. Diese Ungewissheit ist als prädiktive Unsicherheit zu verstehen. Sie ist ein Maß für den erwarteten Fehler einer Modellausgabe. Eine geringe prädiktive Unsicherheit ist ein Anzeichen für eine geringe Abweichung der Modellausgabe $\hat{\mathbf{y}}^*$ vom unbekanntem wahren Funktionswert \mathbf{y}^* . Für Klassifizierungsprobleme gilt, je geringer die prädiktive Unsicherheit, desto größer ist die Wahrscheinlichkeit für die Übereinstimmung von $\hat{\mathbf{y}}^*$ mit \mathbf{y}^* .

2.2.2 Hauptursachen für prädiktive Unsicherheit

Im Prozess der Entwicklung eines DNNs bis zur Anwendung gibt es zahlreiche Quellen, die dessen Approximation von $p(\mathbf{y}^*|\mathbf{x}^*)$ ungenauer machen und somit die Unsicherheit erhöhen. Die laut [GTA+22] häufigsten und schwerwiegendsten Faktoren werden kurz erläutert.

Die meisten Umgebungen in der realen Welt sind sehr variabel und von ständigen Veränderungen beeinflusst. Betroffen sind verschiedene Attribute wie Form und Aussehen physischer Objekte und Temperatur. Fahrzeuge erscheinen beispielsweise bei bedecktem Himmel dunkler. Eine Veränderung der Daten im Vergleich zum Trainingsdatensatz \mathcal{D} wird Distribution-Shift genannt. Ein Distribution-Shift kann zu signifikanten Unterschieden in der Performance des DNNs und somit zu prädiktiven Unsicherheiten führen.

Die Messungen zur Aufnahme der Daten können eine Limitation der tatsächlichen Daten sein, beispielsweise aufgrund einer geringen Bildauflösung oder dem Überspringen nicht ausreichend vorhandener Informationen beim Messen. Prädiktive Unsicherheit kann außerdem durch Rauschen in den Daten wie Sensorrauschen oder fehlerhaftes Labeling der Trainingsdaten verursacht werden. Letzteres verringert die Certainty des Modells bezüglich eigentlich korrekter Prädiktionen im Verlauf des Trainings und lernt im schlimmsten Fall, inkorrekte Prädiktionen für die betroffenen Eingaben auszugeben.

Abhängig von Modellstruktur und Parameterzahl kann ein Modell mehr oder weniger Details über die Trainingsdaten erlernen, weshalb die Modellstruktur einen direkten Einfluss auf die Performance und Unsicherheit des Netzwerks hat. Zudem tendieren DNNs bei Klassifizierungsproblemen dazu, zu selbstsicher in der Softmaxausgabe zu sein, was am Ende dieses Kapitels verdeutlicht wird. Der Trainingsprozess selbst ist stochastisch und führt zu unterschiedlich guten Modellparametrisierungen und somit zu unterschiedlich hohen prädiktiven Unsicherheiten bezüglich gleicher Eingaben. Dieser Effekt wird von den vielen variierbaren Parametern wie dem Optimierer, der Lernrate, der Abbruchbedingung und den Regularisierungstechniken verstärkt.

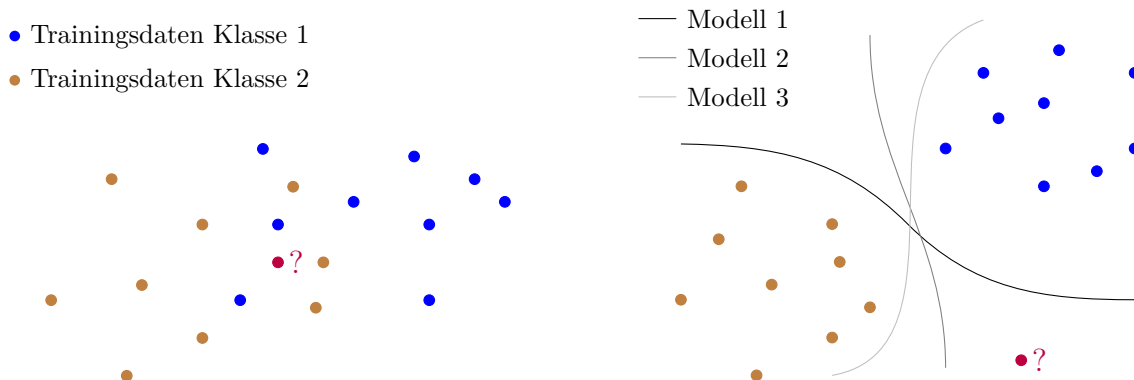
Wenn ein Modell Vorhersagen für Daten treffen soll, die weit von der Domäne der Trainingsdaten D entfernt sind, haben die Prädiktionen eine sehr hohe Unsicherheit. Das Vorliegen solcher, zur Trainingszeit unbekanntener, Out-of-Domain Daten ist teilweise schwer von Methoden zur Uncertainty Estimation zu erkennen.

2.2.3 Klassifizierung der prädiktiven Unsicherheit

Die verschiedenen Ursachen für die Entstehung prädiktiver Unsicherheit motivieren die Unterteilung dieser in verschiedene Kategorien, die einzeln untersucht werden können. J. Gawlikowski et al. [GTA⁺22] unterscheiden zwei mögliche Partitionen der Unsicherheit.

Aleatorische und epistemische Unsicherheit

Die prädiktive Unsicherheit setzt sich aus der Unsicherheit in den Daten (statistische oder aleatorische Unsicherheit) und der Unsicherheit im Modell (systematische oder epistemische Unsicherheit) zusammen. Die epistemische Unsicherheit wird von der mangelnden Kenntnis des neuronalen Netzwerks über die Verteilung der Daten verursacht. Sie kann durch Fehler beim Training, eine nicht ausreichende Modellstruktur, unbekannte Daten oder eine schlechte Konvergenz für den Trainingsdatensatz entstehen. Im Gegensatz zur aleatorischen kann die epistemische Unsicherheit durch Verbesserung von Modellarchitektur, Optimierung des Trainingsprozesses und Vergrößerung des Trainingsdatensatzes vermindert werden. Die aleatorische Unsicherheit resultiert aus der komprimierten Darstellung realer Daten als Eingaben für ein Modell. Beim Repräsentieren von Daten aus der realen Welt in einem Sample gehen Informationen verloren, die für das Modell nützlich sein könnten.



(a) Beispiel für hohe aleatorische Unsicherheit. (b) Beispiel für hohe epistemische Unsicherheit.

Abbildung 2.5: Visualisierung der aleatorischen und epistemischen Unsicherheit am Beispiel eines unbekannteren Samples (roter Datenpunkt) für ein Klassifikationsmodell, das zwei Kategorien unterscheidet. Zur Vereinfachung sind die Datenpunkte lediglich zweidimensional.

In Abbildung 2.5a wird eine Prädiktion mit großer aleatorischer Unsicherheit veranschaulicht. Die Repräsentation des Datenpunkts lässt trotz der Nähe zu den Trainingsdaten keine eindeutige Klassifizierung zu. Die Erfassung der Daten war zu wenig informativ.

Möglicherweise könnte im dreidimensionalen Raum eine Zuordnung mit geringerer aleatorischer Unsicherheit vorgenommen werden.

Die aleatorische Unsicherheit ist in Abbildung 2.5b allgemein sehr gering, da Modelle die Klassen leicht voneinander separieren können. Eine Prädiktion für den betrachteten Datenpunkt hat hier eine hohe epistemische Unsicherheit, da die Modelle beim Training keine ähnlichen Daten klassifiziert haben. Die Kurven in Abbildung 2.5b sind Entscheidungsgrenzen (Decision Boundaries). Sie trennen die Räume ab, in denen das jeweilige Modell Datenpunkte einer bestimmten Klasse zuordnet. Man sieht, dass die Zuordnung des Samples (roter Datenpunkt) nicht eindeutig ist. Die Unsicherheit könnte durch einen besseren Trainingsdatensatz vermindert werden.

In-Domain, Domain-Shift und Out-of-Domain Unsicherheit

Verschiedene Definitionsbereiche der Eingabedaten führen zu unterschiedlich hohen Unsicherheiten. Deshalb wird die prädiktive Unsicherheit oft auch anhand des Definitionsbereiches unterschieden: In-Domain, Domain-Shift und Out-of-Domain Unsicherheit.

In-Domain Unsicherheit ist die Unsicherheit bezüglich Eingabedaten, welche von einer Verteilung stammen, von der angenommen wird, dass sie mit der der Trainingsdaten übereinstimmt. Die Unsicherheit resultiert hier aus mangelndem Wissen über die Verteilung der Trainingsdaten. Dies kann mit epistemischer Unsicherheit zusammenhängen und ist in diesem Fall beispielshalber durch Verbesserung der Trainingsdaten oder des Trainingsprozesses reduzierbar. In-Domain Unsicherheit kann allerdings auch die Komplexität des Problems und somit aleatorische Unsicherheit als Ursache haben [ALMV20].

Die Unsicherheit für Eingabedaten, deren Verteilung eine verschobene Form der Trainingsdatenverteilung ist, wird als Domain-Shift Unsicherheit bezeichnet. Sie wird durch externe und umweltbedingte Faktoren ausgelöst. Eine Domänenverschiebung kann zum Beispiel nach einiger Zeit der Anwendung des Modells aufgrund Veränderungen der realen Welt auftreten. Solche Verschiebungen erhöhen die Unsicherheit häufig, da das neuronale Netzwerk die Daten aus der Domänenverschiebung basierend auf den Trainingsdaten nicht erklären kann. Die Unsicherheit kann teilweise reduziert werden, indem Samples aus verschobenen Verteilungen im Trainingsdatensatz inkludiert werden [OFR⁺19]. Ein Beispiel für eine Domänenverschiebung ist ein verschwommenes Bild eines Hundes für ein Netzwerk, das auf scharfen Tierbildern trainiert wurde. Domain-Shift Unsicherheit kann hier mittels Data Augmentation reduziert werden, was bedeutet, dass eine leicht modifizierte Kopie der Trainingsdaten zum Trainingsdatensatz hinzugefügt wird.

Out-of-Domain (OOD) Unsicherheit repräsentiert die Unsicherheit bezüglich unbekannter Daten. Die Verteilung unbekannter Daten ist weit von der der Trainingsdaten entfernt. DNNs können zwar von Domain-Shift Datensamples In-Domain Wissen extrahieren, jedoch nicht von OOD Daten. OOD Unsicherheit beschreibt zum Beispiel den Fall, dass ein neuronales Netzwerk einen Vogel vorhersagen soll, obwohl es nur trainiert ist Katzen und Hunde zu klassifizieren. Die Unsicherheit entsteht, weil das Netzwerk über kein OOD Wissen verfügt, da es nicht für OOD Daten bestimmt ist.

2.2.4 Zu selbstsichere Softmaxscores für Klassifizierungsprobleme

Die Ausgaben neuronaler Netzwerke für Klassifizierungsaufgaben werden häufig als Klassenwahrscheinlichkeiten interpretiert, was den Anschein erweckt, dass sie direkt als Certainty Estimates verwendet werden können. Durch Anwendung der Softmaxfunktion

$$\text{softmax}(\mathbf{z})_j = \frac{\exp(z_j)}{\sum_{c=1}^C \exp(z_c)} \quad (2.14)$$

auf die Ausgabe \mathbf{z} der letzten Schicht ist die Vorhersage des Netzwerks für jede Klasse j positiv sowie kleiner 1 und $\sum_{c=1}^C \text{softmax}(\mathbf{z})_c = 1$. C steht hier für die Anzahl der Klassen. Dies hat den Anschein einer kategorialen Wahrscheinlichkeitsverteilung.

Die Softmaxausgaben könnten bereits als aleatorische Unsicherheit interpretiert werden, jedoch ist der maximale Softmaxscore meistens zu selbstsicher, was bedeutet, dass neuronale Netzwerke dazu tendieren, der vorhergesagten Klasse eine zu hohe Wahrscheinlichkeit zuzuweisen [GTA⁺22, GPSW17]. Beim Training neuronaler Netzwerke werden Ausgaben, die einen hohen Wert für die korrekte Klasse vorhersagen, mit einem geringen Fehler belohnt. Die Abweichung der Softmaxausgabe vom One-Hot-kodierten Label $(0, \dots, 0, 1, 0, \dots, 0)$ mit dem Maximalwert 1 für die korrekte Klasse ist in diesem Fall geringer. Dies führt dazu, dass der Wert der vorhergesagten Klasse im Softmaxvektor oft höher ist als die Wahrscheinlichkeit der Korrektheit dieser Vorhersage. Daher sollten die Einträge der Softmaxausgabe nicht als Wahrscheinlichkeiten für die Zugehörigkeit einer Eingabe zu den Klassen betrachtet werden.






















							
Pred:	1	1	3	7	7	1	1
Conf:	1	0.8	0.97	1	0.98	0.94	1
							
Pred:	3	8	1	1	8	8	8
Conf:	1	0.62	1	0.84	0.99	1	0.99
							
Pred:	7	7	7	1	5	6	6
Conf:	1	1	0.88	0.86	0.87	0.97	1

Abbildung 2.6: Softmax Prädiktionen trainiert auf dem MNIST-Datensatz für verschiedene Rotationen der Testdaten. Diese Beispiele zeigen, wie eine Domain-Verschiebung zu viel zu selbstsicheren Prädiktionen eines einfachen Klassifizierungsnetzwerks führen kann (Grafik von [GTA⁺22]).

Die epistemische Unsicherheit hingegen ist keinesfalls aus der Softmaxausgabe herleitbar. Dies ist vor allem bei Out-of-Domain Testdaten deutlich bemerkbar. OOD Daten haben eine sehr hohe epistemische Unsicherheit. Ein neuronales Netzwerk, das auf Bildern der zwei Klassen Hund und Katze trainiert wurde, wird höchstwahrscheinlich nicht 50% Hund und 50% Katze ausgeben, wenn es das Bild eines Vogels als Eingabe erhält. Das Netzwerk erkennt bestimmte Merkmale im Bild, die zwar weder richtig zur Katze noch zum Hund passen, aber vielleicht eher zur Katze als zum Hund. Folglich gibt das Netzwerk eine höhere Wahrscheinlichkeit für die Katze aus.

Abb. 2.6 zeigt, wie Spiegelungen und Rotationen einer Ziffer vom MNIST-Datensatz zu falschen Prädiktionen mit hohen Softmaxwerten führt. Es konnte gezeigt werden, dass ein Netzwerk bei Kombination der Aktivierungsfunktion $g(x) = \max(0, x)$ mit der Softmaxfunktion immer sicherer wird, je größer die Distanz zwischen einem Out-of-Domain Datum und dem Trainingsdatensatz ist [HAB19]. Mit der Distanz erhöht sich jedoch die epistemische Unsicherheit, weshalb ein Softmaxscore keinesfalls als Maß für die prädiktive Unsicherheit verwendet werden sollte.

2.3 Methoden zur Uncertainty Estimation in DNNs

Wie bereits in Abschnitt 2.2.2 erläutert, können viele Faktoren epistemische und aleatorische Unsicherheit bei Vorhersagen neuronaler Netzwerke verursachen. Dies verhindert die Exklusion der prädiktiven Unsicherheit in fast allen Anwendungen. Jedoch ist die exakte Berechnung der prädiktiven Unsicherheit eines Deep Neural Networks (DNN) ebenfalls nicht möglich. Die verschiedenen Unsicherheiten können im allgemeinen nicht akkurat modelliert werden oder überhaupt deutlich voneinander unterschieden werden. Demnach ist das Finden und Verbessern von Methoden zur Uncertainty Estimation noch ein großes Thema der Forschung. Dieses Kapitel gewährt einen Einblick in bereits entwickelte Methoden zur Uncertainty Estimation. Ausgewählte Ansätze werden ausführlich vorgestellt und im nachfolgenden Kapitel evaluiert.

2.3.1 Überblick und Einteilung von Methoden zur Uncertainty Estimation

Bisher entwickelte Methoden unterscheiden sich in der Art und Anzahl der verwendeten DNNs. Beispielsweise werden deterministische oder stochastische Netzwerke eingesetzt. J. Gawlikowski et al. [GTA⁺22] unterscheiden vier Kategorien, in welche sich die Methoden einteilen lassen:

Einzelne deterministische Netzwerke berechnen Vorhersagen basierend auf einem einzelnen Feedforward Lauf. Die Unsicherheit einer Prädiktion wird entweder direkt vom Netzwerk oder von einem zusätzlichen externen Netzwerk abgeschätzt.

Bayessche Methoden fassen jegliche Arten stochastischer DNNs zusammen. Dies sind Netzwerke, bei denen zwei Feedforward Läufe mit gleicher Eingabe zu unterschiedlichen Ergebnissen führen.

Ensemble Methoden verwenden die Ausgaben von mehreren verschiedenen und unabhängigen deterministischen Netzwerken bezüglich derselben Eingabe um eine Prädiktion und dessen Unsicherheit zu berechnen.

Testzeit Data Augmentation Methoden nutzen ein einzelnes deterministisches Netzwerk. Es werden mehrere Feedforward Läufe für verschieden augmentierte Versionen einer Testeingabe ausgeführt. Das heißt, die Eingaben des Netzwerks sind leicht modifizierte Kopien des Samples, sodass von der Varietät der Ausgaben eine Aussage über die prädiktive Unsicherheit für dieses Sample getroffen werden kann.

Aufgrund der hohen Anzahl verschiedener Methoden liegt der Fokus der Arbeit auf einzelnen, häufig angewandten Methoden aus verschiedenen Kategorien, welche in diesem Kapitel detailliert erläutert werden. Bei Methoden mit einem einzelnen deterministischen Netzwerk wird in Abschnitt 2.3.5 ein Ansatz mit einem externen Netzwerk für die Uncertainty Estimation betrachtet. T. Ramalho und M. Miranda stellten 2019 ihr externes Netzwerk in ihrem Paper [RM19] unter dem Namen Neighborhood Uncertainty Classifier (NUC) vor. Da die Methoden in dieser Arbeit nur auf Klassifizierungsprobleme angewandt werden und der NUC speziell für Klassifizierungsprobleme entwickelt wurde, ist die Betrachtung dieser Methode sinnvoll für die Tests der Arbeit. Methoden, bei denen das Netzwerk, das die Prädiktion berechnet, gleichzeitig eine Schätzung für die Unsicherheit ausgibt, werden nicht weiter betrachtet. Die untersuchten Ansätze zur Uncertainty Estimation sollen auf verschiedene und bereits trainierte Modelle anwendbar sein. Eine Modifikation jedes einzelnen Modells für die Uncertainty Estimation ist nicht vorgesehen.

Von den Ensemble Methoden werden mehrere ausgewählte Ensembles betrachtet, die in Abschnitt 2.3.4 beschrieben werden. Die bayessche Art Vorhersagen zu berechnen, stellt einen Ansatz bereit, mit dem die wahrscheinlichkeitstheoretische Modellierung der prädiktiven Wahrscheinlichkeit möglich ist [GTA⁺22], was Grundlage für die bayessche Uncertainty Estimation ist. Dieses Vorgehen wird im Folgenden zunächst erklärt, bevor auf eine spezifische Methode zur Uncertainty Estimation, das Monte Carlo Dropout, genauer eingegangen wird. Das Monte Carlo Dropout wurde 2016 erstmals von Y. Gal und Z. Ghahramani in [GG16] erwähnt und ist neben den Ensemble Methoden eine der am häufigsten referenzierten Methoden zur Uncertainty Estimation. Sie ist auf verschiedene Modellarchitekturen anwendbar und approximiert die prädiktive Unsicherheit unter minimalem Implementierungsaufwand.

2.3.2 Bayessche neuronale Netzwerke (BNNs)

Bayessche neuronale Netzwerke (BNNs) kombinieren neuronale Netzwerke mit bayesischem Lernen und bieten eine Alternative zum Lernen mit Maximum Likelihood Methoden. Sie modellieren Wahrscheinlichkeitsverteilungen der Netzwerkparameter, verbessern die Robustheit bezüglich Overfitting, können leicht von kleinen Datensätzen lernen und werden häufig für die Uncertainty Estimation verwendet [Gal16].

Beim bayesischen Modellieren werden für gegebene Eingaben $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$ der Trainingsdaten und ihre Labels $\mathbf{Y} = \{\mathbf{y}_n\}_{n=1}^N$ die Parameter $\boldsymbol{\omega}$ einer Funktion $\mathbf{y} = f_{\boldsymbol{\omega}}(\mathbf{x})$ gesucht, die am ehesten die Labels generiert. Im Folgendem wird stets die Notation $f(\mathbf{x}; \boldsymbol{\omega})$ für die Funktion $f_{\boldsymbol{\omega}}(\mathbf{x})$ genutzt. Bei BNNs handelt es sich bei den Parametern $\boldsymbol{\omega}$ um die Gewichtsmatrizen \mathbf{W}_i und Bias Vektoren \mathbf{b}_i für jede Schicht i . Eine Wahrscheinlichkeitsverteilung über dem Parameterraum wird verwendet. Dies ist die Prior Distribution $p(\boldsymbol{\omega})$. Sie repräsentiert die vorausgehende Annahme, welche Parameter am wahrscheinlichsten die Daten generiert haben, bevor der erste Datenpunkt observiert wurde. Eine gute Annahme zu spezifizieren, ist aktuell noch ein wenig verstandenes Problem [GTA⁺22]. Häufig wird die Standardnormalverteilung $p(\mathbf{W}_i) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ für BNNs verwendet [Gal16].

Des Weiteren muss eine Likelihood Distribution $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega})$ definiert werden. Sie repräsentiert das probabilistische Modell, das von den Eingaben mit gegebenen Parametern $\boldsymbol{\omega}$ die Ausgaben erzeugt. Für Klassifizierungsprobleme ist dies eine Softmax Likelihood:

$$p(y = c|\mathbf{x}, \boldsymbol{\omega}) = \frac{\exp(f_c(\mathbf{x}; \boldsymbol{\omega}))}{\sum_{c'} \exp(f_{c'}(\mathbf{x}; \boldsymbol{\omega}))}. \quad (2.15)$$

Mit f_c ist hier die Netzwerkausgabe des c -ten Neurons vor Anwendung der Softmaxfunktion gemeint. Eine Ausgabe \mathbf{y}^* für eine neue Eingabe \mathbf{x}^* wird bei BNNs mit der Wahrscheinlichkeitsverteilung $p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y})$ durch Integration über alle möglichen Werte der Netzwerkparameter bestimmt:

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) = \int \underbrace{p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega})}_{\text{aleatorisch}} \underbrace{p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})}_{\text{epistemisch}} d\boldsymbol{\omega}. \quad (2.16)$$

Diesen Prozess nennt man Inference³. Inference ermöglicht die Abschätzung der prädiktiven Unsicherheit. Die epistemische Unsicherheit ist beim bayesschen Modellieren als Wahrscheinlichkeitsverteilung der Modellparameter $\boldsymbol{\omega}$ formuliert. Sie ist im hinteren Term, in der sogenannten Posterior Distribution $p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})$ enthalten. Die aleatorische Unsicherheit hingegen wird als Verteilung der Ausgaben \mathbf{y}^* des Modells definiert und ist im vorderen Term, der Likelihood Distribution, zu finden. [GTA⁺22]

Allerdings ist das Integral in Gleichung 2.16 aufgrund der Posterior Distribution meistens nicht berechenbar und muss approximiert werden. Das am häufigsten verwendete Verfahren hierfür ist die Monte Carlo Approximation, auf die in Kapitel 2.3.3 eingegangen wird. Die Posterior Distribution $p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})$ über dem Parameterraum $\boldsymbol{\omega}$ kann mit dem Satz von Bayes bestimmt werden:

$$p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y}) = \frac{p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega})p(\boldsymbol{\omega})}{p(\mathbf{Y}|\mathbf{X})}. \quad (2.17)$$

Die Model Evidence oder marginale Likelihood

$$p(\mathbf{Y}|\mathbf{X}) = \int p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega})p(\boldsymbol{\omega})d\boldsymbol{\omega} \quad (2.18)$$

im Nenner von Gleichung 2.17 ist eine normalisierende Konstante. Um die besten Parameterwerte $\boldsymbol{\omega}$ für einen gegebenen Datensatz zu finden, können Konstanten vernachlässigt werden. Die sogenannte Maximum A Posteriori (MAP) Methode verwendet daher lediglich $p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y}) \propto p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega})p(\boldsymbol{\omega})$. Jedoch ist das Ziel eines BNNs die gesamte Posterior Distribution zu bestimmen, anstatt diese lediglich zu maximieren. Somit ist es BNNs prinzipiell möglich, eine gute Abschätzung der epistemischen Unsicherheit zu erhalten.

Für die exakte Berechnung der Posterior Distribution muss die Model Evidence mittels Gleichung 2.18 ermittelt werden. Das Lösen des Integrals nennt man Marginalisierung der Likelihood über $\boldsymbol{\omega}$. BNNs sind jedoch zu komplex, um die Marginalisierung analy-

³Bemerke, Inference beim bayesschen Modellieren hat eine andere Bedeutung als in Deep Learning. Beim bayesschen Modellieren ist Inference der Prozess der Integration über Modellparameter. Im Gegensatz dazu ist mit Inference im Kontext von Deep Learning meist die Ausführung eines Modells zur Prädiktion auf unbekanntem Daten gemeint [Fen21].

tisch auszuführen. Bei einfachen bayesschen Modellen ist der Prior $p(\boldsymbol{\omega})$ zur Likelihood konjugiert, was bedeutet, dass die Posterior Distribution zur selben Familie von Wahrscheinlichkeitsverteilungen gehört wie der Prior [Sch11]. Dies hat zur Folge, dass das Integral in Gleichung 2.18 analytisch ausgewertet werden kann. Für BNNs ist dies nicht der Fall. Eine Abschätzung wird benötigt. [Gal16]

Es wurden verschiedene Ansätze entwickelt, um die Posterior Distribution und somit die Posterior Inference zu approximieren. Das am häufigsten verwendete Verfahren ist Variational Inference. Anstatt die wahre Posterior Distribution zu bestimmen, wird eine abweichende Verteilung $q_\theta(\boldsymbol{\omega})$ definiert, deren Struktur leicht auszuwerten ist. Sie soll trotzdem möglichst nah an der tatsächlichen Posterior Distribution sein. Um dies zu erreichen, wird die Kullback-Leiber (KL) Divergence in Bezug auf θ minimiert. Die KL Divergence $D_{KL}(P||Q)$ ist ein Maß dafür, wie sehr sich eine Wahrscheinlichkeitsverteilung P von einer zweiten Referenzverteilung Q unterscheidet [Fen21]:

$$D_{KL}(q_\theta(\boldsymbol{\omega})||p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})) = \int q_\theta(\boldsymbol{\omega}) \log \frac{q_\theta(\boldsymbol{\omega})}{p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})} d\boldsymbol{\omega}. \quad (2.19)$$

Hierbei ist zu beachten, dass das Integral nur definiert ist, wenn $q_\theta(\boldsymbol{\omega})$ stetig ist. Gleichung 2.19 ist noch immer nicht berechenbar, da die Posterior Distribution $p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})$ selbst in der Formel vorkommt. Durch Umformungen kann der nicht auswertbare Teil der Posterior Distribution, die Model Evidence, separiert werden:

$$D_{KL}(q_\theta(\boldsymbol{\omega})||p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})) = \int q_\theta(\boldsymbol{\omega}) (\log q_\theta(\boldsymbol{\omega}) - \log p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})) d\boldsymbol{\omega} \quad (2.20)$$

$$\stackrel{2.17}{=} \int q_\theta(\boldsymbol{\omega}) \left(\log q_\theta(\boldsymbol{\omega}) - \log \frac{p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega})p(\boldsymbol{\omega})}{p(\mathbf{Y}|\mathbf{X})} \right) d\boldsymbol{\omega} \quad (2.21)$$

$$= \int q_\theta(\boldsymbol{\omega}) (\log q_\theta(\boldsymbol{\omega}) - \log p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega}) - \log p(\boldsymbol{\omega}) + \log p(\mathbf{Y}|\mathbf{X})) d\boldsymbol{\omega} \quad (2.22)$$

$$= \int q_\theta(\boldsymbol{\omega}) (\log q_\theta(\boldsymbol{\omega}) - \log p(\boldsymbol{\omega})) d\boldsymbol{\omega} - \int q_\theta(\boldsymbol{\omega}) (\log p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega}) - \log p(\mathbf{Y}|\mathbf{X})) d\boldsymbol{\omega} \quad (2.23)$$

$$\stackrel{2.20}{=} D_{KL}(q_\theta(\boldsymbol{\omega})||p(\boldsymbol{\omega})) - \int q_\theta(\boldsymbol{\omega}) (\log p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega}) - \log p(\mathbf{Y}|\mathbf{X})) d\boldsymbol{\omega} \quad (2.24)$$

$$= D_{KL}(q_\theta(\boldsymbol{\omega})||p(\boldsymbol{\omega})) - \int q_\theta(\boldsymbol{\omega}) \log p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega}) d\boldsymbol{\omega} + \log p(\mathbf{Y}|\mathbf{X}) \int q_\theta(\boldsymbol{\omega}) d\boldsymbol{\omega} \quad (2.25)$$

$$\stackrel{4}{=} - \underbrace{\left(\int q_\theta(\boldsymbol{\omega}) \log p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega}) d\boldsymbol{\omega} - D_{KL}(q_\theta(\boldsymbol{\omega})||p(\boldsymbol{\omega})) \right)}_{\text{Evidence Lower Bound (ELBO)}} + \log p(\mathbf{Y}|\mathbf{X}). \quad (2.26)$$

Der fordere Term, die negative ELBO Funktion, ist für bestimmte q_θ berechenbar und ihre Maximierung entspricht der Minimierung der KL Divergence $D_{KL}(q_\theta(\boldsymbol{\omega})||p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y}))$, wie durch Umstellung von Gleichung 2.26 ersichtlich:

$$D_{KL}(q_\theta(\boldsymbol{\omega})||p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})) = -ELBO(q_\theta) + \log p(\mathbf{Y}|\mathbf{X}) \quad (2.27)$$

$$ELBO(q_\theta) = \log p(\mathbf{Y}|\mathbf{X}) - D_{KL}(q_\theta(\boldsymbol{\omega})||p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})) \quad (2.28)$$

$$\Rightarrow ELBO(q_\theta) \leq \log p(\mathbf{Y}|\mathbf{X}). \quad (2.29)$$

⁴In Gleichung 2.25 gilt $\int q_\theta(\boldsymbol{\omega}) d\boldsymbol{\omega} = 1$, da $q_\theta(\boldsymbol{\omega})$ eine Dichtefunktion ist, weshalb $\int q_\theta(\boldsymbol{\omega}) d\boldsymbol{\omega}$ als Faktor vor dem Logarithmus der Model Evidence $\log p(\mathbf{Y}|\mathbf{X})$ in Gleichung 2.26 entfällt.

Durch Maximierung von ELBO wird folglich ein $q_\theta(\boldsymbol{\omega})$ gefunden, welches die Posterior Distribution gut approximiert. Die möglichen Verteilungen für $q_\theta(\boldsymbol{\omega})$ werden oft bereits vor der Maximierung eingeschränkt. Man wählt eine Menge an Verteilungen, sodass die Berechnung oder Abschätzung von ELBO leicht fällt.

Unter Verwendung der approximierten Posterior Distribution ergibt sich basierend auf Gleichung 2.16 eine Abschätzung für die prädiktive Wahrscheinlichkeitsverteilung:

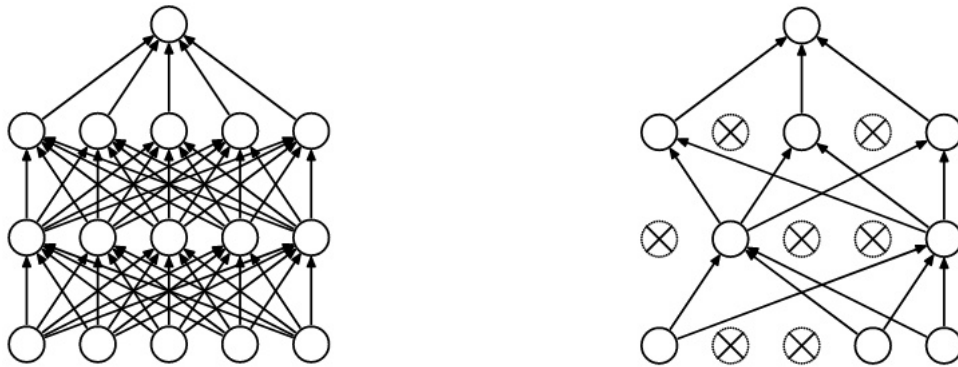
$$p(\mathbf{y}^* | \mathbf{x}^*, \mathbf{X}, \mathbf{Y}) \approx \int p(\mathbf{y}^* | \mathbf{x}^*, \boldsymbol{\omega}) q_\theta(\boldsymbol{\omega}) d\boldsymbol{\omega} =: q_\theta^*(\mathbf{y}^* | \mathbf{x}^*). \quad (2.30)$$

2.3.3 Monte Carlo Dropout Methode

Die Monte Carlo Dropout (MC Dropout) Methode ist ein bayesscher Ansatz zur Uncertainty Estimation. Sie interpretiert neuronale Netzwerke, die mit Dropout Regularisierung trainiert wurden, als BNNs [Fen21] und wurde von Y. Gal [Gal16] vorgestellt.

Dropout in neuronalen Netzwerken

Um Overfitting bei neuronalen Netzwerken zu reduzieren, wird oft die stochastische Regularisierungstechnik Dropout beim Trainieren der Netzwerke verwendet. Die Idee ist, zufällig einzelne Neuronen mit einer Dropout-Rate von p_{dropout} zu entfernen, sodass aus einem Netzwerk verschiedene Teilnetzwerke mit gleichen Neuronen aber verringerter Kapazität entstehen. Die Teilnetzwerke sollen eigenständig lernen, die richtigen Prädiktionen zu erzeugen [GBC17]. Abb. 2.7 visualisiert diesen Ansatz. Das vollständige Netzwerk vereinigt nach dem Training das Wissen seiner Teile und verfügt allgemein über eine höhere Performance als es ohne Dropout Training der Fall sein würde.



(a) Gewöhnliches neuronales Netzwerk

(b) Netzwerk nach der Anwendung von Dropout

Abbildung 2.7: Vereinfachte Darstellung von Dropout in einem neuronalen Netzwerk mit zwei verborgenen Schichten. Mit einem Kreuz gekennzeichnete Neuronen wurden entfernt (Grafik von [SHK⁺14]).

Ein effizientes Vorgehen zur Entfernung eines Neurons ist, den Ausgabewert Null zu erzeugen. Hierfür wird für jeden Feedforward Durchlauf des Netzwerks ein Vektor mit binären Masken $\boldsymbol{\mu}^{(l)}$ für eine Schicht l als Stichprobe

$$\mu_j^{(l)} \sim \text{Bernoulli}(p_{\text{dropout}}) \quad (2.31)$$

entnommen. Anschließend erfolgt die Multiplikation dieses Vektors mit den Ausgaben der Neuronen der l -ten Schicht. Somit wird festgelegt, welche Neuronen im aktuellen Teilnetzwerk verwendet werden. Bei Anwendung des Netzwerks zur Testzeit werden wieder alle Neuronen verwendet und deren Ausgaben mit p_{dropout} multipliziert. Die Ausgabe eines Neurons zur Testzeit entspricht somit der erwarteten Ausgabe zur Trainingszeit. [SHK⁺14]

Monte Carlo Sampling

Monte Carlo Sampling ist eine Methode zur Approximation von Summen oder Integralen, die nicht exakt berechenbar sind. Die Summe oder das Integral wird als Erwartungswert einer bestimmten Verteilung betrachtet. Seien

$$s = \sum_{\mathbf{x}} q(\mathbf{x})f(\mathbf{x}) = \mathbb{E}_q[f(\mathbf{x})] \quad (2.32)$$

oder

$$s = \int q(\mathbf{x})f(\mathbf{x})d\mathbf{x} = \mathbb{E}_q[f(\mathbf{x})] \quad (2.33)$$

die Summe oder das Integral, das approximiert werden soll, und f eine beliebige Funktion. Die Umformungen 2.32 und 2.33 können nur unter der Bedingung, dass q bei der Summe eine diskrete Wahrscheinlichkeitsverteilung und beim Integral eine Dichtefunktion ist, angewandt werden. Des Weiteren sollte q so definiert sein, dass von ihr leicht Stichproben entnommen werden können. [GBC17]

Monte Carlo Sampling approximiert den Erwartungswert unter Anwendung des Gesetzes der großen Zahlen mit einem korrespondierenden Mittelwert. Es werden m Stichproben $\mathbf{x}_1, \dots, \mathbf{x}_m$ von q entnommen und der empirische Mittelwert

$$\bar{s}_m = \frac{1}{m} \sum_{i=1}^m f(\mathbf{x}_i) \quad (2.34)$$

gebildet. Zum Beispiel kann so das Integral von Gleichung 2.30 approximiert werden:

$$q_{\theta}^*(\mathbf{y}^*|\mathbf{x}^*) = \int p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega})q_{\theta}(\boldsymbol{\omega})d\boldsymbol{\omega} = \mathbb{E}_{q_{\theta}}[p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega})] \quad (2.35)$$

$$\approx \frac{1}{m} \sum_{i=1}^m p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega}_i) \quad \text{mit } \boldsymbol{\omega}_i \sim q_{\theta}(\boldsymbol{\omega}). \quad (2.36)$$

Die Prädiktion $\hat{\mathbf{y}}^*$ wird als Mittelwert der Ausgaben von m deterministischen Netzwerken f_1, \dots, f_m , parametrisiert mit $\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_m$, berechnet [GTA⁺22]:

$$\hat{\mathbf{y}}^* \approx \frac{1}{m} \sum_{i=1}^m f_i(\mathbf{x}^*; \boldsymbol{\omega}_i). \quad (2.37)$$

Dropout als bayessche Approximation

Die Monte-Carlo Dropout Methode ermöglicht die Verwendung von Dropout in neuronalen Netzwerken zur Approximation von Inference. Dropoutschichten sind als Bernoulli-

verteilte Zufallsvariablen formuliert und das Training eines Netzwerks mit Dropoutschichten führt annähernd Variational Inference aus [GTA⁺22].

Dieser Ansatz definiert eine approximierende Verteilung $q_\theta(\boldsymbol{\omega})$ für die Posterior Distribution $p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})$ über Matrizen $\boldsymbol{\omega} = \{\mathbf{W}_i\}_{i=1}^L$. Hier sei L die Anzahl der Schichten des Netzwerks und \mathbf{W}_i die Parametermatrix der i -ten Schicht mit Dimensionen $K_i \times K_{i-1}$. Jede Schicht hat eine grundlegende Gewichtsmatrix \mathbf{M}_i , die vom aktuellen Dropoutvektor $\boldsymbol{\mu}^{(i)}$ unabhängig ist. Es gilt

$$q_\theta(\mathbf{W}_i) = \mathbf{M}_i \cdot \text{diag}([\mu_j^{(i)}]_{j=1}^{K_{i-1}}) \quad (2.38)$$

$$\mu_j^{(i)} \sim \text{Bernoulli}(p_i) \text{ für } i = 1, \dots, L, j = 1, \dots, K_{i-1}. \quad (2.39)$$

Die zu erlernenden Parameter sind $\theta = \{\mathbf{M}_i\}_{i=1}^L$. Bayessche neuronale Netzwerke minimieren den negativen Evidence Lower Bound (ELBO) aus Gleichung 2.26, um $q_\theta(\boldsymbol{\omega})$ der Posterior Distribution anzunähern:

$$\mathcal{L}(\theta|\mathbf{X}, \mathbf{Y}) = D_{KL}(q_\theta(\boldsymbol{\omega})||p(\boldsymbol{\omega})) - \int q_\theta(\boldsymbol{\omega}) \log p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega}) \quad (2.40)$$

$$= D_{KL}(q_\theta(\boldsymbol{\omega})||p(\boldsymbol{\omega})) - \mathbb{E}_{q_\theta}[\log p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega})]. \quad (2.41)$$

Der 2. Term kann als Summe über die einzelnen Trainingsbeispiele

$$- \sum_{n=1}^N \mathbb{E}_{q_\theta}[\log p(\mathbf{y}_n|\mathbf{x}_n, \boldsymbol{\omega})] \quad (2.42)$$

formuliert werden, wobei N die Anzahl der Trainingsdaten ist. Jeder Term in der Summe wird durch Monte Carlo Sampling mit einem einzigen Parametervektor $\boldsymbol{\omega}_n \sim q_\theta(\boldsymbol{\omega})$ approximiert. Man erhält eine erwartungstreue Schätzung für die Log-Likelihood $\log p(\mathbf{y}_n|\mathbf{x}_n, \boldsymbol{\omega})$:

$$- \sum_{n=1}^N \mathbb{E}_{q_\theta}[\log p(\mathbf{y}_n|\mathbf{x}_n, \boldsymbol{\omega})] \approx - \sum_{n=1}^N \log p(\mathbf{y}_n|\mathbf{x}_n, \boldsymbol{\omega}_n) \text{ mit } \boldsymbol{\omega}_n \sim q_\theta(\boldsymbol{\omega}). \quad (2.43)$$

Dieses Vorgehen entspricht einer Epoche beim Training eines Netzwerks, das Dropout enthält. Für jeden Feedforward Durchlauf des Netzwerks, werden die Zufallsvektoren $\boldsymbol{\mu}$ für das Dropout erzeugt. Hieraus ergibt sich der Parametervektor $\boldsymbol{\omega}_n$. Die Likelihood Distribution ist für Klassifizierungsprobleme eine Softmax Likelihood, weshalb die negative Log-Likelihood hier durch Minimierung des Cross Entropy Loss minimiert werden kann:

$$\min_{\theta} \left\{ - \sum_{n=1}^N \log p(\mathbf{y}_n|\mathbf{x}_n, \boldsymbol{\omega}_n) \right\} \Leftrightarrow \min_{\theta} \left\{ \frac{1}{N} \sum_{n=1}^N E(\mathbf{y}_n, f(\mathbf{x}_n; \boldsymbol{\omega}_n)) \right\}. \quad (2.44)$$

Um Overfitting zu verringern, werden der Lossfunktion meistens Regularisierungsterme beim Optimieren der Parameter eines neuronalen Netzwerks hinzugefügt. Dabei handelt es sich häufig um die L2-Regularisierung $\|\mathbf{W}_i\|_2^2 = \sum_{j=1}^{K_i} \sum_{k=1}^{K_{i-1}} w_{i,j,k}^2$ mit einer beliebigen Gewichtung λ (hier vereinfacht ohne Bias Vektoren dargestellt):

$$\mathcal{L}_{\text{dropout}}(\theta|\mathbf{X}, \mathbf{Y}) := \frac{1}{N} \sum_{n=1}^N E(\mathbf{y}_n, f(\mathbf{x}_n; \boldsymbol{\omega}_n)) + \lambda \sum_{i=1}^L \|\mathbf{W}_i\|_2^2. \quad (2.45)$$

Sie bezweckt, dass sich der absolute Betrag der Gewichte verringert und sich dadurch die Verteilung der Gewichte einer Normalverteilung mit Erwartungswert Null annähert. Der Unterschied zwischen $q_\theta(\boldsymbol{\omega})$ und dem beliebigen Prior $p(\boldsymbol{\omega}) = \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$ verringert sich folglich, weshalb die Minimierung des L2-Terms nahezu äquivalent zur Minimierung der KL-Divergenz von approximativer Posterior und Prior Distribution $D_{KL}(q_\theta(\boldsymbol{\omega})||p(\boldsymbol{\omega}))$ ist. Somit ist gezeigt, dass durch Minimierung der Lossfunktion von Gleichung 2.45 die negative ELBO Funktion eines BNNs aus Gleichung 2.40 minimiert wird. Ein Netzwerk mit Dropoutschichten sowie L2-Regularisierung beim Training kann folglich als BNN interpretiert werden. [GG16, Gal16]

Abschätzung der prädiktiven Unsicherheit

Zur Bestimmung der Unsicherheit einer Prädiktion \mathbf{y}^* wird die approximative prädiktive Wahrscheinlichkeitsverteilung $q_\theta^*(\mathbf{y}^*|\mathbf{x}^*)$ benötigt. Wie bereits erwähnt, ermöglicht Monte Carlo Sampling eine Abschätzung dieser basierend auf Gleichung 2.30:

$$q_\theta^*(\mathbf{y}^*|\mathbf{x}^*) = \int p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega})q_\theta(\boldsymbol{\omega})d\boldsymbol{\omega} \quad (2.46)$$

$$\approx \frac{1}{T} \sum_{t=1}^T p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega}_t) \quad \text{mit} \quad \boldsymbol{\omega}_t \sim q_\theta(\boldsymbol{\omega}). \quad (2.47)$$

Im Bezug auf Dropout entspricht dies dem Mittelwert von T stochastischen Feedforward Durchläufen eines Netzwerks mit Dropout zur Testzeit. Die Abschätzung

$$\mathbb{E}_{q_\theta^*(\mathbf{y}^*|\mathbf{x}^*)}(\mathbf{y}^*) \approx \frac{1}{T} \sum_{t=1}^T f(\mathbf{x}^*; \boldsymbol{\omega}_t) \quad (2.48)$$

bezeichnet man als MC Dropout Schätzer [GG16]. Jede Ausgabe $f(\mathbf{x}^*; \boldsymbol{\omega}_t)$ wird mit aktivierten Dropoutschichten erzeugt, wobei die Parameterstichprobe $\boldsymbol{\omega}_t$ von der approximativen Posterior Distribution $q_\theta(\boldsymbol{\omega})$ gezogen wird. Die Stichprobe bestimmt, welche Neuronen in der aktuellen Feedforward Operation beibehalten werden. Ein größeres T resultiert in einer besser approximierten prädiktiven Wahrscheinlichkeitsverteilung. In der Praxis wurde gezeigt, dass Konvergenz nach 30 bis 50 Dropout Durchläufen zu erwarten ist [Fen21].

Bei Klassifizierungsproblemen sind die Netzwerkausgaben meistens Softmaxscores \hat{s}^* und die Likelihood für die c -te Klasse und den t -ten Dropout Feedforward Lauf kann ausgedrückt werden als $p(y^* = c|\mathbf{x}^*, \boldsymbol{\omega}_t) = \hat{s}_{t,c}^*$. Die prädiktive Wahrscheinlichkeit für Klasse c wird durch den Mittelwert der einzelnen Softmaxscores abgeschätzt:

$$p(y^* = c|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) \approx \frac{1}{T} \sum_{t=1}^T \hat{s}_{t,c}^*. \quad (2.49)$$

Abb. 2.8 visualisiert das Vorgehen zur Ermittlung von Zentral- und Streuungsmaßen der approximierten prädiktiven Wahrscheinlichkeit, wie beispielsweise Mittelwert und Standardabweichung. Diese können entweder direkt als Uncertainty Estimates oder von bestimmten Metriken verwendet werden. Letzteres ist vor allem bei Klassifizierungsproblemen üblich. Die Metriken werden in Kapitel 2.4 vorgestellt.

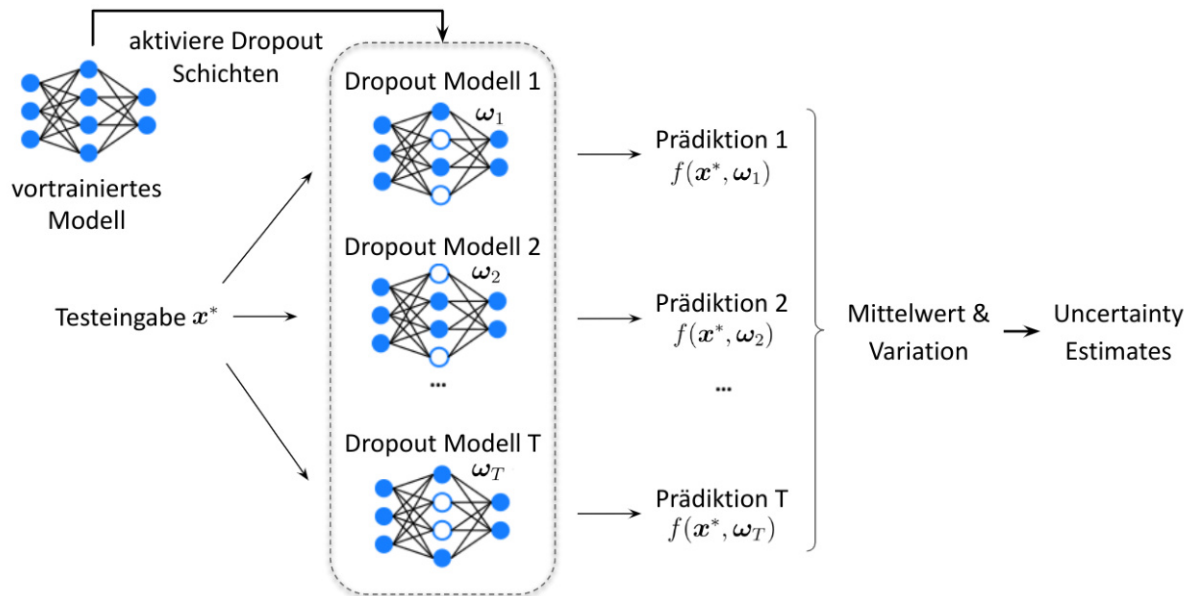


Abbildung 2.8: Visualisierung der MC Dropout Uncertainty Estimation. Das Netzwerk durchläuft T Feedforward Operationen mit aktiven Dropoutschichten für eine Testeingabe x^* . Die einzelnen Ausgaben werden verwendet, um die Statistiken für die prädiktive Unsicherheit zu ermitteln. (Grafik angelehnt an [Fen21])

Zusammenfassend bietet die MC Dropout Methode einen praktischen Ansatz, um approximierte Inference in bayesschen neuronalen Netzwerken durchzuführen. So kann die prädiktive Unsicherheit von bereits mit Dropout trainierten neuronalen Netzwerken durch Aktivierung des Dropouts zur Testzeit abgeschätzt werden. Leider werden mehrere Dropout Durchläufe je Prädiktion zur Testzeit benötigt, weshalb diese Art der Uncertainty Estimation in sicherheitskritischen Echtzeitsystemen noch kaum anwendbar ist.

2.3.4 Ensemble Methoden

Ensemble Methoden sind weitere Ansätze, die mittels Ausgaben unterschiedlicher Modelle die prädiktive Unsicherheit abschätzen. Sie trainieren mehrere Modelle separat. Ausgaben werden aus den Prädiktionen der einzelnen sogenannten Ensemblemitglieder generiert. Die Motivation hierfür ist zunächst, unabhängig von der Uncertainty Estimation, dass eine Gruppe von Modellen im allgemeinen weniger Fehler macht als ein einzelnes Modell und somit die Performance verbessert. [GBC17]

B. Lakshminarayanan et al. [LPB17] haben entdeckt, dass das Kombinieren von Prädiktionen mehrerer Netzwerke die prädiktive Wahrscheinlichkeit gut approximiert. Sei M die Anzahl der Netzwerke eines Ensembles und seien $\{\omega_m\}_{m=1}^M$ ihre Parameter. Für die Testdaten x^* gibt jedes Netzwerk im Ensemble eine Prädiktion von $p(\mathbf{y}^* | x^*, \omega_m)$ für den Zielwert \mathbf{y}^* aus. Die prädiktive Wahrscheinlichkeitsverteilung des Ensembles kann mittels des Durchschnitts der Ausgaben der Mitglieder approximiert werden [Fen21]:

$$p(\mathbf{y}^* | x^*, \mathbf{X}, \mathbf{Y}) \approx \frac{1}{M} \sum_{m=1}^M p(\mathbf{y}^* | x^*, \omega_m). \quad (2.50)$$

Dieses Vorgehen wird Model Averaging genannt [GBC17]. Die Gleichung ist ähnlich zur Gleichung 2.47 der Monte Carlo Dropout Methode, da MC Dropout selbst als Ensemble von Netzwerken mit aktivem Dropout interpretiert werden kann [LPB17]. Allerdings gehören Ensembles allgemein nicht zu den bayesschen Methoden und Gleichung 2.50 wird als uniform-gewichtetes Mixture-Modell betrachtet. Ein größeres M führt zu besseren Uncertainty Estimates, jedoch wurde gezeigt, dass ein Ensemble von fünf Netzwerken ausreichend ist, um gute Abschätzungen der Unsicherheit zu erhalten [LPB17]. Dies impliziert einen deutlich verringerten Rechenaufwand im Vergleich zum MC Dropout, bei dem meistens 30 bis 50 Prädiktionen für ein Uncertainty Estimate ermittelt werden. Allerdings ist der Speicherplatzbedarf auch fünfmal höher.

Bei Klassifizierungsproblemen wird ebenfalls ähnlich wie beim Monte Carlo Dropout der Durchschnitt der Softmaxausgaben verschiedener Mitglieder gebildet:

$$p(y^* = c | \mathbf{x}^*, \mathbf{X}, \mathbf{Y}) \approx \frac{1}{M} \sum_{m=1}^M \hat{s}_{m,c}^*, \quad (2.51)$$

wobei \hat{s}_m^* der Softmax Score des m -ten Netzwerks des Ensembles ist [Fen21].

Ein großer Unterschied zu anderen Methoden für Uncertainty Estimation ist die Anzahl der lokalen Optima, die für die Berechnung einer Prädiktion berücksichtigt werden. Neuronale Netzwerke definieren eine sehr nicht-lineare Abbildung, sodass die Lossfunktion viele lokale Optima enthält. Beim Training konvergiert ein deterministisches Netzwerk zu einem dieser Optima. Bayessche neuronale Netzwerke konvergieren ebenfalls zu einem Optimum, aber berücksichtigen zusätzlich die Unsicherheit an diesem lokalen Optimum, indem dessen nahe Umgebung durch Variation der Parameter ω betrachtet wird. Im Gegensatz dazu bestehen Ensemble Methoden aus mehreren Netzwerken, die sehr wahrscheinlich zu unterschiedlichen lokalen Optima konvergieren. Abbildung 2.9 visualisiert dieses Verhalten. Das Ziel der sogenannten Multi-Mode Evaluation ist, dass verschiedene lokale Optima zu Modellen mit unterschiedlichen Stärken und Schwächen in den Prädiktionen führen können. Das Zusammenspiel dieser Modelle verbessert die Gesamtleistung des Ensembles und hat ebenfalls Vorteile für die Uncertainty Estimation, denn damit falsche Prädiktionen erkannt werden können, sollten sich die Mitglieder bei unsicheren Ausgaben unterschiedlich verhalten. [GTA⁺22, FHL19]

Das Erreichen einer möglichst hohen Varietät zwischen den einzelnen Netzwerken ist eine der größten Herausforderungen bei Ensemble Methoden. Hierfür gibt es verschiedene Ansätze, um die Ensemblemitglieder zu erzeugen und zu trainieren. Häufig verwendet werden die zufällige Initialisierung der Modellparameter, das Mischen der Trainingsdaten, Bagging, Boosting, Data Augmentation und das Einsetzen verschiedener Modellarchitekturen.

Aufgrund der vielen lokalen Minima der Lossfunktion von Deep Neural Networks, führen unterschiedliche Initialisierungen der Parameter zu verschiedenen Resultaten. Beim Training mit Minibatches ergeben unterschiedliche Reihenfolgen der Daten ebenfalls andere Endwerte der Parameter. Folglich ist es üblich jedes Ensemble Mitglied zufällig zu initialisieren und beim Training das Mischen der Daten zu aktivieren. [GTA⁺22]

Bagging und Boosting sind zwei Ensemble Methoden, die die Verteilung der Trainingsdaten pro Modell verändern, indem neue Datenmengen als Stichproben vom Trainingsdatensatz entnommen werden. Bei Bagging werden die Daten von einer stetigen Gleich-

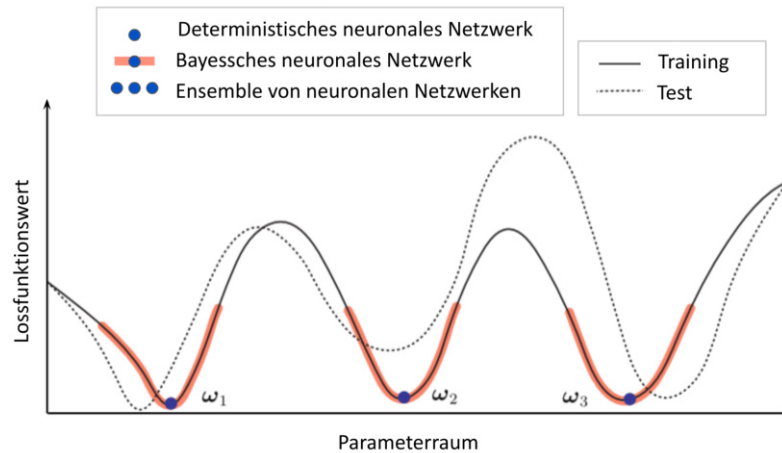


Abbildung 2.9: Darstellung der verschiedenen Evaluierungsverhalten von deterministischen neuronalen Netzwerken, BNNs und einem Ensemble unterschiedlicher deterministischer neuronaler Netzwerke. Zur Vereinfachung der Visualisierung ist der Parameterraum eindimensional. BNNs betrachten die Umgebung des Parametervektors, während das Ensemble verschiedene Parameterwerte lernt, die zur Testzeit unterschiedlich gute Prädiktionen liefern (Grafik von [GTA⁺22]).

verteilung über den Trainingsdaten mit Zurücklegen entnommen. Somit sind einzelne Trainingsbeispiele gar nicht und andere mehrfach in der Stichprobe enthalten. Dieser Samplingprozess wird für jedes Mitglied im Ensemble wiederholt, welche anschließend auf den Stichproben trainiert werden. Die unterschiedlichen Verteilungen in diesen neuen Datensätzen führen zu Modellen, die sich verschieden verhalten. Beim Boosting werden die Ensemblemitglieder nacheinander trainiert. Die Wahrscheinlichkeit für die Auswahl eines Trainingsbeispiels für das nächste Modell ist abhängig von der Performance des bisher trainierten Ensembles. Somit ist das Training bei Boosting im Vergleich zu Bagging um die Anzahl der Ensemblemitglieder langsamer, da die Modelle nicht parallel trainiert werden können. [GTA⁺22]

Bei der Anwendung von Data Augmentation auf die Eingabedaten, werden diese leicht variiert, beispielshalber gedreht, gespiegelt oder verschoben. Die zufällige Data Augmentation des Trainingsdatensatzes pro Ensemble Mitglied führt zu kleinen Differenzen in den Eingaben der einzelnen Modelle und somit zu Variationen der Mitglieder. Leicht ersichtlich ist ebenfalls, dass verschiedene Modellarchitekturen eine hohe Diversität zwischen den Ensemblemitgliedern erzeugen, da jede Architektur zu einer anderen Loss-Verteilung führt.

Ensemble Methoden, die in den folgenden Kapiteln evaluiert werden, sind Bagging und Data Augmentation, wobei die verwendeten Trainingsdaten zusätzlich gemischt und die Modellparameter zufällig initialisiert werden. Dies erhöht die Variation der Ensemblemitglieder weiter. Generell ist das Mischen der Daten beim Training von Deep Neural Networks üblich, denn es führt unter anderem zu einer besseren Generalisierung. Auch die zufällige Initialisierung der Gewichte mit Stichproben einer Wahrscheinlichkeitsverteilung ist ein verbreiteter Standard. Da Boosting Bagging sehr ähnelt, aber aufgrund der fehlenden Parallelisierbarkeit einen höheren Zeitaufwand impliziert, wird Boosting im Rahmen dieser Bachelorarbeit nicht zu implementieren.

Der Fokus der Arbeit liegt, neben dem Vergleich ausgewählter Methoden zur Uncertainty Estimation, auf der Verwendung der Uncertainty Estimates für das aktive Lernen. Hier wird die Unsicherheit eines vortrainierten Modells f für eine Menge neuer Daten abgeschätzt. Anschließend werden die unsichersten Daten gelabelt und zum Trainingsdatensatz hinzugefügt. Von den unsicheren Prädiktionen ist der höchste Lerneffekt beim erneuten Training zu erwarten (siehe Kapitel 4). Dementsprechend sollen sich die Uncertainty Estimates auf die Ausgaben dieses spezifischen Modells beziehen, weshalb die Verwendung verschiedener Modellarchitekturen, die von der des Netzwerks f abweichen, für die Ensemblemitglieder nicht sinnvoll erscheint.

Ensemble Methoden haben wie der Monte Carlo Dropout Ansatz den Vorteil, dass sie leicht anwendbar sind. Die Mitglieder werden unabhängig voneinander trainiert, was die Parallelisierung des Trainingsprozesses sowie die Erweiterung des Ensembles um zusätzliche Modelle ermöglicht. Jedoch wachsen Rechenaufwand und Speicherplatzbedarf für Training und Anwendung des Ensembles linear mit der Anzahl der Mitglieder. Ensemble Methoden sind folglich kaum anwendbar, wenn Rechen- und Speicherkapazität begrenzt sind und zeitkritische Anwendungen vorliegen [Fen21]. Ansätze wie Pruning versuchen Ensemble Methoden effizienter zu machen, indem redundante Mitglieder entfernt werden [GTA⁺22]. Solche Ansätze werden in dieser Arbeit nicht weiter betrachtet, da die Auswahl neuer Trainingsdaten für das aktive Lernen keine zeitkritische Anwendung ist.

2.3.5 Neighborhood Uncertainty Classifier

Der Neighborhood Uncertainty Classifier (NUC) gehört zu den deterministischen Methoden zur Uncertainty Estimation und wurde 2019 von T. Ramalho et al. [RM19] vorgestellt. Der Classifier ist ein externes Netzwerk, das die Unsicherheit der Vorhersagen eines Hauptnetzwerks schätzt. Hierfür wird die Distanz zwischen einer Eingabe und den nächstgelegenen Nachbarn des Trainingsdatensatzes abgeschätzt, da diese einen hohen Einfluss auf die Korrektheit und somit auf die Certainty einer Prädiktion haben kann.

Da die meisten Datensätze sehr hochdimensional sind, ist die direkte Approximation der Dichte der Datenverteilung mit zu hohem Rechenaufwand verbunden. T. Ramalho et al. [RM19] lösen dieses Problem, indem die Dichte im high-level Repräsentationsraum eines neuronalen Netzwerks bestimmt wird. Der high-level Repräsentationsraum kann als komprimierte Repräsentation der Daten betrachtet werden. Annahme ist, dass die Distanz im Repräsentationsraum eines gut trainierten Modells in Korrelation zur Distanz in der Verteilung der Daten steht. Im Paper [RM19] wird diese Annahme bestätigt.

Der NUC verwendet den Repräsentationsraum vor der letzten Schicht des Hauptmodells. Sei \mathbf{x} eine Eingabe, L die Anzahl an Schichten und $\mathbf{r}_i = f^i(\mathbf{x})$ die Ausgabe der i -ten Schicht. Der letzte Repräsentationsvektor \mathbf{r}_{L-1} wird nur noch linear abgebildet, um die finale Ausgabe vor Anwendung der Softmaxfunktion zu erhalten. Die \mathbf{r}_i haben eine geringere Dimension als \mathbf{x} und von der letzten Repräsentation \mathbf{r}_{L-1} , ab jetzt lediglich mit \mathbf{r} bezeichnet, ist zu erwarten, dass Datenpunkte gleicher Klassen grob gruppiert zusammen liegen. Dies ermöglicht die Verwendung von k Nearest Neighbors zur approximativen Berechnung der Dichte des Trainingsdatensatzes.

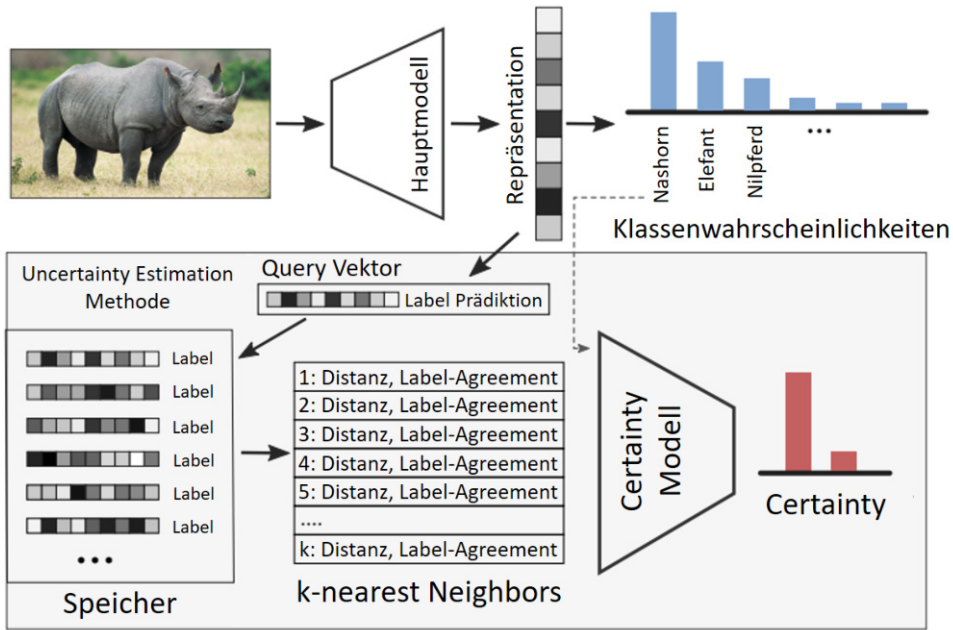


Abbildung 2.10: Schematische Darstellung der Abschätzung der Certainty mittels Density Estimation im Repräsentationsraum von [RM19].

Nützliche Informationen, die aus den k Nearest Neighbors zu gewinnen sind, sind die Distanz zum Repräsentationsvektor \mathbf{r} der Eingabe, dem sogenannten Query Vektor, und das Label des jeweiligen Nachbarn. Aus den Labels und der vorhergesagten Klasse \hat{y} für \mathbf{x} kann geschlussfolgert werden, wie viele der Nachbarn mit dieser Klasse übereinstimmen. Diese Informationen dienen als Eingabe des Certainty Netzwerks. Der Parameter k bestimmt von wie vielen Nachbarn die Informationen bestimmt und verwendet werden. Unterschiedliche k führen möglicherweise zu verschiedenen guten Ergebnissen des Certainty Netzwerks. Dessen Performance in Abhängigkeit von k wird in Kapitel 3.3.2 untersucht.

Abbildung 2.10 visualisiert die Verwendung des Neighborhood Uncertainty Classifiers (NUC). Das Hauptmodell, ein Deep Neural Network, berechnet Klassenwahrscheinlichkeiten für einen bekannten Trainingsdatensatz. Der high-level Repräsentationsvektor direkt vor der Ausgabeschicht des Netzwerks und das Label werden für jeden Datenpunkt gespeichert. Für neue Testbeispiele \mathbf{x}^* wird ebenfalls der Query Vektor berechnet und mit den gespeicherten Vektoren verglichen, um die Informationen von den nächstgelegenen Nachbarn für das Certainty Modell zu ermitteln. Das Certainty Modell schätzt auf Basis dieser Informationen die Wahrscheinlichkeit für die Korrektheit der Prädiktion \hat{y}^* des Hauptmodells ab.

Die Implementierung des Certainty Netzwerks g_θ dieser Bachelorarbeit orientiert sich an der Modellarchitektur und dem Trainingsalgorithmus des Papers [RM19]. Der Certainty Score bezüglich einer Eingabe \mathbf{x} des Hauptmodells ist definiert als:

$$c(\mathbf{x}) = g_\theta\left(\{(d_{(j)}, \mathbb{I}(y_{(j)} = \hat{y})\}_{j \in \mathcal{N}(\mathbf{r})}, s(\hat{y})\right). \quad (2.52)$$

Hierbei bezeichnet $y_{(j)}$ das Label des j -ten Nachbarn und $s(\hat{y})$ ist der Softmaxscore der vorhergesagten Klasse, folglich der maximale Wert der Softmaxausgabe. Als Maß für die Distanz $d_{(j)}$ zwischen dem Repräsentationsvektor \mathbf{r} der Eingabe und dem des j -ten Nach-

barn wird die euklidische Distanz verwendet. Die Funktion g_θ ist ein Deep Feedforward Neural Network, welches mit binärem Cross Entropy Loss

$$L_{CE}(c, t) = - \sum_{n=1}^N (t_n \log(c(\mathbf{x}_n)) + (1 - t_n) \log(1 - c(\mathbf{x}_n))). \quad (2.53)$$

trainiert wird. Die binären Labels sind $t_i = 1$ falls $y_i = \hat{y}_i$, sonst $t_i = 0$. Die letzte Schicht des Certainty Modells hat ein Neuron mit einer Sigmoidfunktion $g(z) = (1 + e^z)^{-1}$ als Aktivierungsfunktion für die binäre Klassifizierung.

Algorithmus 1 Trainingsprozedur des Neighborhood Uncertainty Classifiers

Require: k Anzahl Nachbarn, \mathcal{A} die Menge aller Repräsentationen, θ Modellparameter, f trainiertes Hauptmodell
for Epoche \in Anzahl Epochen **do**
 for $\mathbf{x}_n \in$ Trainingsdatensatz **do**
 $\mathbf{r}_n \leftarrow f^{L-1}(\mathbf{x}_n)$
 $\hat{y}_n \leftarrow \operatorname{argmax} f^L(\mathbf{x}_n)$
 $t_n \leftarrow (y_n = \hat{y}_n)$
 $\{(d_{(j)}, y_{(j)})\}_{j \in \mathcal{N}(\mathbf{r}_n)} \leftarrow \operatorname{knn}(\mathbf{r}_n, k, \mathcal{A})$
 $c(\mathbf{x}_n) \leftarrow g_\theta(\{(d_{(j)}, \mathbb{I}(y_{(j)} = \hat{y}_n)\}_{j \in \mathcal{N}(\mathbf{r}_n)}, s(\hat{y}_n))$
 $\theta_{new} \leftarrow \theta_{old} + \lambda \nabla_{\theta} L(c(\mathbf{x}_n), t_n)$
 end for
end for

Die gesamte Trainingsprozedur ist in Algorithmus 1 zusammengefasst. Benötigt wird ein bereits trainiertes Hauptmodell f , was für jede Eingabe \mathbf{x} die Vorhersage \hat{y} und den zugehörigen high-level Repräsentationsvektor \mathbf{r} berechnet. \mathcal{A} ist die Menge der gespeicherten high-level Repräsentationen aller Trainingsdaten. Hierbei ist wichtig, dass f den Trainingsdatensatz nicht perfekt klassifiziert, damit auch Daten mit dem Label $t = 0$ für das Training des Certainty Netzwerks vorhanden sind. Dies ist üblicherweise der Fall, da die meisten Datensätze ausreichend komplex sind und Regularisierungstechniken in den Modellen Overfitting verhindern.

Der NUC eignet sich besonders für Out-of-Domain (OOD) Erkennung. OOD Daten haben eine hohe Distanz zu den Trainingsdaten, weshalb die Nachbarn im Repräsentationsraum eines gut trainierten Netzwerks auch eine hohe Distanz zum Datenpunkt haben. Das Certainty Netzwerk kann leicht eine große Distanz mit hoher Unsicherheit verknüpfen und somit Prädiktionen der OOD Beispiele als unsicher identifizieren. Die eigene Implementierung wird in Kapitel 3.3.3 für OOD Erkennung getestet.

2.4 Messung der prädiktiven Unsicherheit bei Klassifizierungsproblemen

Während der Neighborhood Uncertainty Classifier einen einzelnen Wert für die Unsicherheit einer Prädiktion berechnet, resultiert die Anwendung von Monte Carlo Dropout und Ensemble Methoden zunächst nicht in einem interpretierbaren Certainty Score. Um hier

für Klassifizierungsprobleme ein Maß für die Unsicherheit zu erhalten, werden verschiedene Metriken verwendet. Einige schätzen hauptsächlich die epistemische und andere eher die aleatorische Unsicherheit ab.

Prädiktive Entropie

Eine häufig verwendete Metrik für die aleatorische und die epistemische Unsicherheit ist die Entropie der prädiktiven Wahrscheinlichkeit [Fen21]. Sei \mathcal{D} der Trainingsdatensatz mit $\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\}$, C die Anzahl verschiedener Klassen, \mathbf{x} eine Modelleingabe und y das zugehörige Label. Y. Gal [Gal16] definiert die Prädiktive Entropie als

$$\mathcal{H}_p(y|\mathbf{x}, \mathcal{D}) = - \sum_{c=1}^C p(y = c|\mathbf{x}, \mathcal{D}) \log(p(y = c|\mathbf{x}, \mathcal{D})). \quad (2.54)$$

Dabei ist $p(y|\mathbf{x}, \mathcal{D})$ die kategoriale prädiktive Wahrscheinlichkeitsverteilung. Sie ordnet jeder Klasse die Wahrscheinlichkeit, die korrekte Prädiktion zu sein, zu. Der Monte Carlo Dropout Ansatz und die Ensemble Methoden ermitteln eine Approximation dieser Wahrscheinlichkeitsverteilung für Klassifizierungsprobleme mittels der Gleichungen 2.49 beziehungsweise 2.51 (siehe Kapitel 2.3.3 und 2.3.4).

Entropie misst die Unvorhersagbarkeit eines Systems. Eine große Entropie spricht für ähnliche Klassenwahrscheinlichkeiten im Klassenvektor p und eine geringe Entropie deutet darauf hin, dass eine Klasse deutlich hervorsteht. Das Modell ist sich im letzten Fall sicherer bei der Prädiktion dieser Klasse. Die Entropie erreicht ihren Minimalwert Null, wenn sich das Netzwerk bei der Prädiktion eindeutig sicher ist, also $p(y = c|\mathbf{x}, \mathcal{D}) = 1$ für eine Klasse c , was in der Realität nicht exakt erreichbar ist. Maximale Entropie steht für maximale Unsicherheit und wird mit der Prädiktion $p(y = c|\mathbf{x}, \mathcal{D}) = \frac{1}{C}$ für alle Klassen c erreicht. [Fen21]

Softmaxentropie

Neben der prädiktiven Wahrscheinlichkeitsverteilung $p(y|\mathbf{x}, \mathcal{D})$ kann auch eine einzelne Netzwerkausgabe $p(y = c|\mathbf{x}, \boldsymbol{\omega})$ als Uncertainty Estimate verwendet werden. Dies bedeutet, dass die Softmaxausgabe $\hat{\mathbf{s}}$ eines Netzwerks bereits ohne die zusätzliche Anwendung einer rechenintensiven Methode wie das MC Dropout zur Uncertainty Estimation verwendet werden kann. Die Softmaxausgabe $\hat{\mathbf{s}}$ ermöglicht ebenfalls die Berechnung einer Entropie. Analog zu Formel 2.54 ist die Softmaxentropie folgendermaßen definiert [Fen21]:

$$\mathcal{H}_s(y|\mathbf{x}, \boldsymbol{\omega}) = - \sum_{c=1}^C p(y = c|\mathbf{x}, \boldsymbol{\omega}) \log(p(y = c|\mathbf{x}, \boldsymbol{\omega})) = - \sum_{c=1}^C \hat{s}_c \log(\hat{s}_c). \quad (2.55)$$

Die Softmaxentropie ist vielmehr ein Maß für die aleatorische als für die epistemische Unsicherheit. Im Gegensatz zur prädiktiven Entropie erfasst sie somit nicht die gesamte prädiktive Unsicherheit, was ein Nachteil der einzelnen Softmaxausgabe gegenüber Methoden wie dem MC Dropout ist. Jedoch hat die epistemische Unsicherheit oft einen geringen Einfluss auf die Softmaxentropie. Die Größe dieses Einflusses, kann von einer einzelnen Ausgabe $\hat{\mathbf{s}}$ allgemein nicht eingeschätzt werden. [GTA⁺22]

Mutual Information

Das MC Dropout und die Ensemble Methoden ermöglichen neben der Abschätzung der gesamten prädiktiven Unsicherheit durch die prädiktive Entropie auch die Abschätzung der reinen epistemischen Unsicherheit. Eine beliebte Metrik für die epistemische Unsicherheit ist die Mutual Information [GTA⁺22]:

$$\mathcal{I}(y, \boldsymbol{\omega} | \mathbf{x}, \mathcal{D}) = \mathcal{H}_p(y | \mathbf{x}, \mathcal{D}) - \mathbb{E}_{p(\boldsymbol{\omega} | \mathcal{D})}[\mathcal{H}_s(y | \mathbf{x}, \boldsymbol{\omega})]. \quad (2.56)$$

Der erste Term von Gleichung 2.56 ist die prädiktive Entropie aus Gleichung 2.54. Der zweite Term ist der Erwartungswert der bedingten Entropie bezüglich der Netzwerkparameter $\boldsymbol{\omega}$. Dies ist die Softmaxentropie eines Netzwerks, parametrisiert durch $\boldsymbol{\omega}$, aus Gleichung 2.55.

Beim Monte Carlo Dropout Ansatz wird die Softmaxentropie mittels der stochastischen Ausgabe eines einzelnen Feedforward Durchlaufs mit aktivierten Dropoutschichten ermittelt. Anschließend wird der Erwartungswert über die Resultate mehrerer Feedforward Läufe approximiert:

$$\mathbb{E}_{p(\boldsymbol{\omega} | \mathcal{D})}[\mathcal{H}_s(y | \mathbf{x}, \boldsymbol{\omega})] = -\mathbb{E}_{p(\boldsymbol{\omega} | \mathcal{D})} \left[\sum_{c=1}^C p(y = c | \mathbf{x}, \boldsymbol{\omega}) \log(p(y = c | \mathbf{x}, \boldsymbol{\omega})) \right] \quad (2.57)$$

$$\approx -\frac{1}{T} \sum_{t=1}^T \left(\sum_{c=1}^C p(y = c | \mathbf{x}, \boldsymbol{\omega}_t) \log(p(y = c | \mathbf{x}, \boldsymbol{\omega}_t)) \right), \quad (2.58)$$

wobei T für die Anzahl Dropout Durchläufe steht und $\boldsymbol{\omega}_t$ die jeweiligen Netzwerkparameter bestimmt. Die einzelnen Ausgaben der Feedforward Läufe sind nun selbst Zufallsvariablen, dessen Variationen ausgewertet werden. [Fen21]

Für Ensemble Methoden lässt sich die Mutual Information sehr ähnlich berechnen. Hier wird die bedingte Entropie von der Softmaxausgabe eines einzelnen Ensembles bestimmt. Anschließend erfolgt die Approximation des Erwartungswerts durch das arithmetische Mittel der bedingten Entropien mehrerer Mitglieder. [Fen21]

Das Minimum der Mutual Information ist 0 und ein höherer Wert impliziert eine hohe epistemische Unsicherheit [Fen21]. Je größer der Wert, desto stärker weichen die Ausgaben der einzelnen Modelle oder Dropout Durchläufe $p(y | \mathbf{x}, \boldsymbol{\omega})$ von der durchschnittlichen prädiktiven Klassenwahrscheinlichkeit $p(y | \mathbf{x}, \mathcal{D})$ ab.

Anwendungsbeispiel

Die Unterschiede in den Uncertainty Estimates von prädiktiver Entropie, Softmaxentropie und Mutual Information werden am folgenden Beispiel von Y. Gal [Gal16] deutlich. Es sind drei Eingaben eines binären Klassifizierungsproblems gegeben. Für jede Eingabe werden mehrere stochastische Feedforward Durchläufe von Monte Carlo Dropout ausgeführt beziehungsweise mehrere Ausgaben von unterschiedlichen Ensemblemitgliedern berechnet. Folgende Softmaxausgaben werden erhalten:

1. alle Ausgaben haben den Softmaxscore 1 für dieselbe Klasse, z.B. $\{(1, 0), \dots, (1, 0)\}$,
2. alle Ausgaben sind für jede Klasse 0.5: $\{(0.5, 0.5), \dots, (0.5, 0.5)\}$,

3. die Hälfte der Ausgaben hat einen Softmaxscore von 1 für die erste Klasse und die andere Hälfte für die zweite Klasse, z.B. $\{(1, 0), (0, 1), (1, 0), \dots, (0, 1)\}$.

Von jedem einzelnen dieser Softmaxausgaben kann die Softmaxentropie bestimmt werden. Die prädiktive Entropie und die Mutual Information hingegen werden für die Gesamtheit aller Softmaxausgaben eines Beispiels bestimmt.

Die Prädiktion für die Eingabe des ersten Beispiels hat eine hohe Certainty, während die anderen beiden Beispiele eine hohe Unsicherheit aufweisen. Die Metriken prädiktive Entropie, Softmaxentropie und Mutual Information ergeben alle, angewandt auf das erste Beispiel, erwartungsgemäß den Wert 0.

Allerdings unterscheiden sich die Werte für das 2. und 3. Beispiel. Der Durchschnitt aller Ausgaben approximiert die prädiktive Wahrscheinlichkeitsverteilung $p(y|\mathbf{x}, \mathcal{D})$ und ist in beiden Fällen $(0.5, 0.5)$. Die prädiktive Entropie ist somit für beide Beispiele maximal mit $\mathcal{H}_p(y|\mathbf{x}, \mathcal{D}) = -0.5 \cdot \log(0.5) - 0.5 \cdot \log(0.5) = 0.69$. Betrachtet man lediglich die einzelnen Softmaxausgaben, fällt auf, dass die Softmaxentropien für das 2. Beispiel ebenfalls mit einem Wert von 0.69 maximal sind. Jedoch sind die Entropien der Softmaxvektoren beim 3. Beispiel stets 0, da $\mathcal{H}_s(y|\mathbf{x}, \boldsymbol{\omega}) = -0 \cdot \log(0) - 1 \cdot \log(1) = 0$. Im Gegensatz dazu ist die Mutual Information für das 2. Beispiel 0, also minimal, und nur für das 3. Beispiel maximal.

Um diese Differenzen zu verstehen, muss beachtet werden, dass Beispiel 2 und 3 unterschiedliche Unsicherheiten darstellen. Die einzelnen Ausgaben sind im 2. Fall identisch, was bedeutet, dass sich die Gesamtheit der verschieden parametrisierten Modelle sicher bei den erzeugten Softmaxscores sind. Die zugrunde liegende Unsicherheit ist die aleatorische Unsicherheit in den Daten. Da die Mutual Information diese nicht erfasst, ist sie hier korrekter Weise Null.

Der 3. Fall ist ein Beispiel für epistemische Unsicherheit, denn die verschieden parametrisierten Modelle weisen starke Differenzen in ihren Ausgaben auf, die durch weiteres Training vermindert werden können. Hier ist keine aleatorische Unsicherheit zu erkennen, da eine einzelne Softmaxausgabe sehr eindeutig auf eine Klasse hinweist. Die prädiktive Entropie und die Mutual Information können die epistemische Unsicherheit erfassen, die Entropie einer einzelnen Ausgabe in diesem Fall jedoch nicht.

3 Bewertung der Uncertainty Estimation bei Klassifizierungsproblemen

Die im vorigen Kapitel vorgestellten Methoden werden nun auf Klassifizierungsnetzwerke angewandt und anhand verschiedener Kriterien miteinander verglichen. Die für den Vergleich verwendeten Datensätze und Modelle sowie das Training werden zunächst beschrieben. Es folgen verschiedene Qualitätstests, die häufig in wissenschaftlichen Arbeiten durchgeführt wurden. Das Python-Projekt ist auf GitHub unter dem Link <https://github.com/HannaLicht/uncertainty-estimation> einsehbar.

3.1 Verwendete Datensätze und Modelle

Da die Uncertainty Estimation bei Volkswagen für Computer Vision genutzt werden soll, werden Bilddatensätze für den Vergleich verwendet. Convolutional Neural Networks (CNNs) eignen sich besonders gut zur Erkennung von Mustern in Bilddaten [ON15], weshalb der Vergleich der Methoden für CNNs erfolgt.

Im Smart.Production:Lab bei Volkswagen wird das EfficientNet-B3 für Klassifizierungsprobleme verwendet und gehört daher zu den betrachteten Modellen dieses Kapitels. CNNs der EfficientNet-Modellfamilie sind durch ein bestimmtes Skalierungsverhältnis von Tiefe und Breite besonders effizient [TL20]. Die EfficientNet-B3 Modellarchitektur hat 12 Millionen Parameter, weshalb dessen prädiktive Unsicherheit für einen entsprechend komplexen Datensatz abgeschätzt werden sollte.

Es werden die öffentlich zugänglichen Datensätze Cifar10, Cifar100 und Cars betrachtet. Aufgrund strenger Geheimhaltungsrichtlinien können keine internen Daten von Volkswagen verwendet werden. Cifar10 und Cifar100 bestehen jeweils aus einem Trainingsdatensatz mit 50000 Bildern und einem Testdatensatz mit 10000 Bildern. Bei Cifar10 unterscheidet man zwischen 10 und bei Cifar100 zwischen 100 unterschiedliche Klassen, unter anderem Fahrzeuge und Tiere [Kri09]. Mit einer Bildauflösung von 32x32 Pixeln sind diese Datensätze jedoch zu kompakt für das Efficient-NetB3, weshalb sie in dieser Arbeit für kleinere CNNs verwendet werden. Der Cars Datensatz enthält 8144 Trainingsbilder und 8041 Testbilder von Autos. Die Autos sind 196 Klassen bezüglich der Marke, des Modells und des Baujahrs zugeordnet [KSDF13]. Die Bildauflösung ist deutlich höher als die von Cifar10 und Cifar100 und somit gut für das komplexe EfficientNet geeignet. Für die Tests wird eine Skalierung der Bilder von 300x300 Pixel genutzt.

Damit der Monte Carlo Dropout (MC Dropout) Ansatz angewandt werden kann, müssen die CNNs Dropoutschichten enthalten. Dies ist für die EfficientNet Architektur bereits

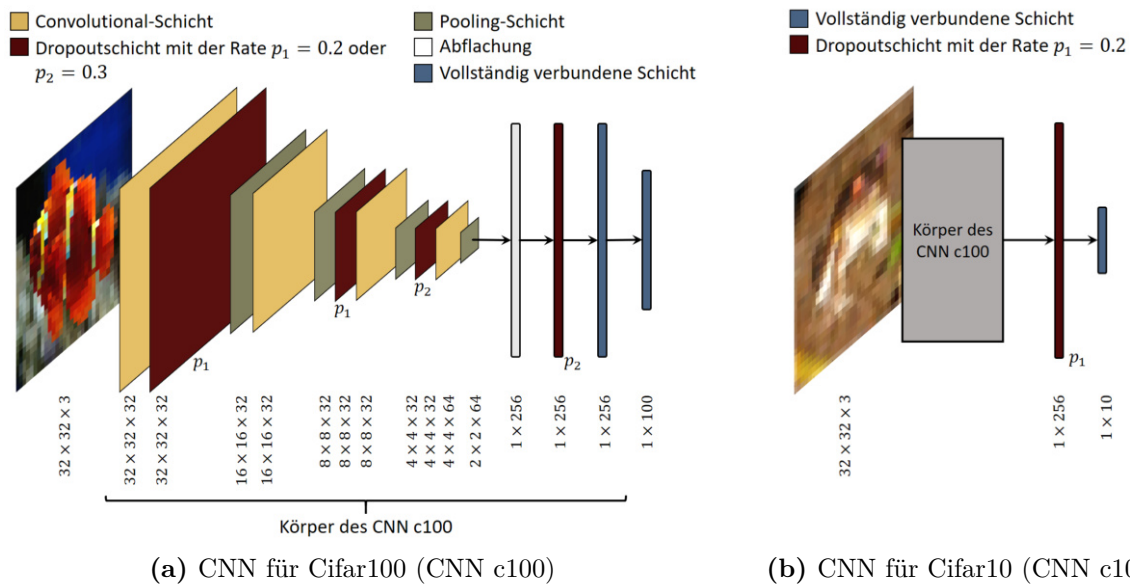


Abbildung 3.1: Architektur der CNNs für die Datensätze Cifar10 und Cifar100.

der Fall. Das Modell verwendet 19 Dropout-Schichten mit einer Dropout-Rate von bis zu 0.2 [TL20]. In die Architektur der verwendeten CNNs für den Cifar100 und Cifar10 Datensatz sind ebenfalls Dropout-Schichten eingebaut. Der Aufbau ist in Abbildung 3.1 visualisiert.

Für das vollständige Training des EfficientNets sind aufgrund der vielen Parameter sehr viele Daten notwendig, um eine hinreichend gute Performance zu erhalten. Das Training ist auf diese Weise sehr aufwendig. Um das zu umgehen, wird für große Netzwerke fast immer Transfer-Lernen verwendet, so auch bei Volkswagen. Transfer-Lernen erzielt die Performanceverbesserung eines Modells auf der Ziel-Domäne durch den Transfer von Wissen über ähnliche Domänen [ZQD⁺20]. Häufig werden die Gewichte eines Modells, das für eine ähnliche Aufgabe entwickelt wurde, als Ausgangspunkt für das eigentliche Training verwendet. Man entfernt die Ausgangsschicht und ergänzt wenige Schichten. Beim anschließenden Training mit Daten aus der Ziel-Domäne werden zunächst nur die Gewichte der ergänzten Schichten verändert. Die Motivation hinter diesem Vorgehen ist, dass nützliche Merkmale (Features) von den bereits vortrainierten Schichten des Modells der ähnlichen Domäne aus Bildern der Ziel-Domäne extrahiert werden. Sie sind der Ausgangspunkt für die Verarbeitung der letzten Schichten. Mit diesen Informationen über die Eingabe gelingt es, sogar mit wenigen Daten der Ziel-Domäne, die letzten Schichten gezielt zu trainieren und eine gute Performance zu erhalten. Ist die Lossfunktion konvergiert, wird erneut mit einer deutlich kleineren Lernrate trainiert. Diesmal werden alle Modellparameter aktualisiert, um sie noch besser an die Ziel-Domäne anzupassen. [ZQD⁺20]

Im Smart.Production:Lab bei Volkswagen verwendet man die Gewichte des auf dem ImageNet Datensatz vortrainierten EfficientNets für das Transfer-Lernen. Der ImageNet Datensatz enthält circa 50 Millionen beschriftete Bilder verschiedener Klassen [DDS⁺09] und eignet sich daher perfekt für das Training großer Netzwerke und als Startpunkt für das Transfer-Lernen. Da keine internen Daten von Volkswagen für diese Arbeit verwendet werden können, wird stattdessen der Cars Datensatz als Ziel-Domäne genutzt wird. Da ImageNet unter anderem Klassen wie Limousine, Cabrio, SUV, Sportwagen und Liefer-

wagen hat, gibt es Parallelen zur Ziel-Domäne von Cars. Dies ist eine gute Voraussetzung für das Transfer-Lernen.

Neben dem kompletten Cifar10 Datensatz werden für die Tests Modelle betrachtet, die nur auf Ausschnitten des Cifar10 Trainingsdatensatz trainiert wurden. Unterschieden wird zwischen der Nutzung von 100, 1000 und 10000 Bildern, um zu untersuchen, wie sich die Performance der Methoden zur Uncertainty Estimation für wenig trainierte DNNs verändert. Die Qualität von Uncertainty Estimates dieser Modelle ist vor allem für das aktive Lernen in Kapitel 4 von Interesse, weil beim aktiven Lernen zu Beginn nur wenige Trainingsdaten vorliegen. Da kaum ein Netzwerk allein mit 100 oder 1000 Daten von Grund auf neu trainiert wird, wird wie für das EfficientNet Transfer-Learning genutzt. Ausgangspunkt des Trainings ist das CNN für den Cifar100 Datensatz, weshalb die Architektur des CNNs für Cifar10 in Abbildung 3.1b dessen Körper beinhaltet. Zwischen den Klassen von Cifar10 und Cifar100 gibt es viele Parallelen, sodass Cifar100 für das Transfer-Lernen verwendet werden kann.

Des Weiteren werden für die Qualitätstests neben Trainings- und Testdatensätzen auch Validierungsdatensätze benötigt. Hierfür bilden die letzten 20% der jeweiligen Trainingsdaten die Validierungsdatensätze. Zusammengefasst werden in den Tests für Cifar10 und Cifar100 jeweils ein Trainingsdatensatz mit 40000, ein Validierungsdatensatz mit 10000 und ein Testdatensatz mit 10000 Bildern verwendet. Für den Cars Datensatz bestehen Trainings-, Validierungs- und Testdatensatz aus 6576, 1568 und 8041 Bildern, unter denen die 196 Klassen in etwa gleichverteilt sind. Für die Netzwerke, die nur auf Ausschnitten des Cifar10 Trainingsdatensatzes trainiert sind, ist der Validierungsdatensatz entsprechend kleiner, sodass die Größen von Trainings- und Validierungsdatensatz im Verhältnis 4 zu 1 stehen.

Erfolgt beim Training der Modelle über einige Epochen keine Performanceverbesserung, wird die Lernrate reduziert. Durch Early Stopping nach noch längerem Fernbleiben einer Verbesserung wird Overfitting vermindert. Tabelle 3.1 fasst die Performance der Modelle nach dem Training zusammen. Trotz weiterer Regularisierungstechniken wie Dropout-schichten und L2-Regularisierung ist für einige Netzwerke eine große Spanne zwischen Trainings- und Testaccuracy zu beobachten. Erzielt werden jedoch keine perfekten Test-accuracys der Modelle. Vielmehr sollen genügend Bilder falsch klassifiziert werden, damit ausreichend Daten vorliegen, für die eine hohe prädiktive Unsicherheit zu erwarten ist.

	CNN c10 ₁₀₀	CNN c10 ₁₀₀₀	CNN c10 ₁₀₀₀₀	CNN c10	CNN c100	Eff Cars
Train	93.0%	94.1%	90.7%	87.9%	61.2%	99.9%
Test	36.7%	54.3%	66.2%	73.7%	40.4%	76.8%

Tabelle 3.1: Accuracys der CNNs auf den Trainings- und Testdatensätzen
(Abkürzungen: c10 - Cifar10; c100 - Cifar100; c10_x - x Trainingsdaten von Cifar10; Eff - EfficientNet-B3)

Die in der Tabelle 3.1 verwendeten Abkürzungen für die sechs Modelle werden für die folgenden Tests durchgehend verwendet. Beispielsweise ist mit CNN c10₁₀₀ das auf 100 Bildern des Cifar10 Trainingsdatensatzes trainierte CNN gemeint.

3.2 Umsetzungsdetails der betrachteten Methoden

Der Vergleich erfolgt für die in Kapitel 2.3 detailliert erläuterten Methoden: das MC Dropout (Abschnitt 2.3.3), die Ensemble Methoden Bagging und Data Augmentation (Abschnitt 2.3.4) sowie der Neighborhood Uncertainty Classifier (Abschnitt 2.3.5). Neben diesen Methoden zur Uncertainty Estimation, wird auch die Performance der einfachen Softmaxausgabe der sechs in Tabelle 3.1 aufgeführten CNNs untersucht. Das ist mittels der in Abschnitt 3.3.2 vorgestellten Softmaxentropie möglich. Für die Anwendung der Ensemble Methoden und des MC Dropouts auf Klassifizierungsnetzwerke werden die Metriken prädiktive Entropie und Mutual Information genutzt.

Für die Mitglieder der Ensemble Methoden wird stets die Modellarchitektur des untersuchten Netzwerks verwendet, um die Ensembles angemessen mit den anderen Methoden vergleichen zu können. Zum Beispiel besteht das Ensemble für den Cars Datensatz aus mehreren EfficientNet-B3-Architekturen, die ebenfalls mit Transfer-Lernen trainiert wurden. Da fünf Ensemblemitglieder nach [LPB17] ausreichend sind, um qualitative Uncertainty Estimates zu erhalten, bestehen die Ensembles für die Tests aus fünf Mitgliedern. Die Veränderung der Trainingsbilder der Mitglieder des Data Augmentation Ensembles erfolgt durch Rotation, Translation und horizontale Spiegelung. Für das MC Dropout sind 30 bis 50 Feedforward Durchläufe eines Netzwerks empfehlenswert [Gal16]. Für die nachfolgenden Untersuchungen werden stets 50 Durchläufe ausgeführt, um möglichst genau die Konvergenz der Methode zu erreichen. Wie viele Nachbarn für die eigene Implementierung des Neighborhood Uncertainty Classifiers am besten geeignet sind, wird in Kapitel 3.3.2 untersucht. Das verwendete Certainty Netzwerk ist ein vollständig verbundenes Netzwerk mit drei Schichten. Aufgrund der geringen Dimension der Eingabedaten des Certainty Netzwerks ist keine komplexe Netzwerkstruktur von Nöten.

3.3 Evaluierungstests der Methoden zur Uncertainty Estimation

Die im Folgenden durchgeführten Tests untersuchen die Performance der Methoden bezüglich verschiedener Kriterien. Im Fokus steht die Interpretierbarkeit der Uncertainty Estimates, die Frage, wie gut korrekte und inkorrekte Prädiktionen anhand des Uncertainty Estimates vorhergesagt werden können, und ob Prädiktionen für Out-of-Domain Daten als unsicher erkannt werden. Für viele Anwendungen ist auch die Betrachtung der Laufzeit von hoher Bedeutung.

3.3.1 Kalibrierungstest - Interpretierbarkeit der Uncertainty Estimates

Eine gute Kalibrierung der Uncertainty Estimates trägt dazu bei, neuronale Netzwerke zuverlässiger zu machen. Sie erhöht die Interpretierbarkeit der Uncertainty Estimates [GPSW17]. Mit einer guten Kalibrierung ist gemeint, dass die Schätzungen die tatsächliche Unsicherheit in den Vorhersagen eines Netzwerks so genau wie möglich widerspiegeln

[Fen21]. Die berechnete Certainty¹ soll der tatsächlichen Wahrscheinlichkeit für die Korrektheit der Prädiktion entsprechen [GPSW17, GTA⁺22]. Dies erleichtert die Einschätzung, ob der Prädiktion vertraut werden kann. Das Netzwerk selbst kann im Fall eines hohen Uncertainty Estimates entscheiden, keine Aussage zu treffen anstatt eine unsichere Vorhersage auszugeben.

Ein nützliches Tool zur Veranschaulichung der Kalibrierungsqualität ist der Calibration Plot [Fen21]. Hierfür werden zunächst Vorhersagen $\hat{\mathbf{y}}^*$ eines Modells für eine Reihe von Beobachtungen $\hat{\mathbf{x}}^*$ gemacht und deren Certainty \hat{p} mittels einer beliebigen Methode abgeschätzt. Im Idealfall sollte ein ganzer Datensatz verwendet werden. Anschließend erfolgt die Gruppierung von Prädiktionen ähnlicher Certainty in T Intervallen mit den Schwellenwerten $0 < \hat{p}_1 < \dots < \hat{p}_t < \dots < \hat{p}_T$ [Fen21]. Mit B_t ist im Folgenden die Menge der Testbeispiele zwischen \hat{p}_{t-1} und \hat{p}_t gemeint. Für jedes Intervall wird der Anteil korrekter Vorhersagen, die Accuracy $\text{acc}(B_t)$, ermittelt. Bei einer guten Kalibrierung sollte beispielsweise gelten, dass von den Prädiktionen mit einem Certainty Score von 70% auch circa 70% korrekt sind [GTA⁺22]. Die horizontale Achse des Calibration Plots repräsentiert die vorhergesagten Certainties des Modells $\text{cert}(B_t)$ und die vertikale Achse die empirische Wahrscheinlichkeit $\text{acc}(B_t)$.

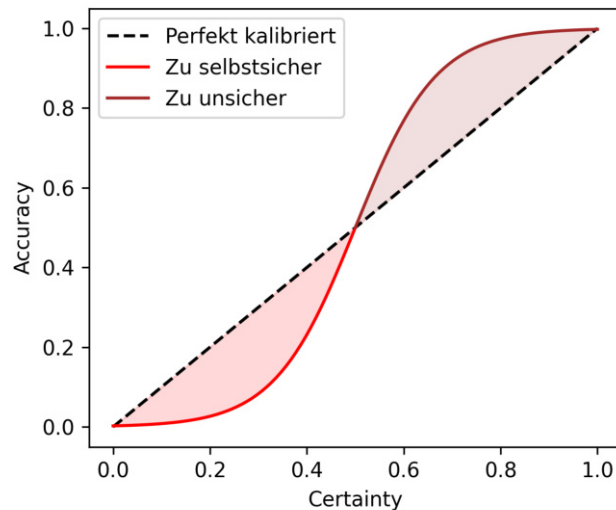


Abbildung 3.2: Beispiel eines Calibration Plots. Die ideale Kalibrationskurve ist eine Gerade durch den Koordinatenursprung mit Anstieg 1. Eine Kurve oberhalb der Diagonale weist auf zu unsichere Certainty Estimates hin, während eine Kurve unterhalb der Diagonale ein Indikator für zu selbstsichere Estimates ist (Abbildung nach [Fen21]).

Gut kalibrierte Certainty Estimates weisen eine Diagonale im Calibration Plot auf, wie in Abbildung 3.2 zu sehen ist. Die Diagonale impliziert, dass die vorhergesagten Certainties stets mit der empirischen Wahrscheinlichkeit für die Korrektheit der Prädiktionen übereinstimmen. Schlecht kalibrierte Werte können dazu tendieren, zu optimistisch oder zu pessimistisch zu sein. Als zu optimistisch, beziehungsweise zu selbstsicher, werden Prädiktionen bezeichnet, deren geschätzte Certainty höher als die tatsächlich zu erwartende

¹Bei der Kalibrierung werden häufig Certainty Estimates anstatt Uncertainty Estimates betrachtet. Certainty Estimates sind jedoch leicht in Uncertainty Estimates umwandelbar und anders herum.

Korrektheit ist. Die Kurve im Calibration Plot liegt unter der Diagonalen. Zu pessimistisch oder zu unsicher sind Prädiktionen, bei denen die vorhergesagte Certainty geringer als die tatsächlich zu erwartende Korrektheit ist. Die Kurve befindet sich in diesem Fall oberhalb der Diagonalen, was Abbildung 3.2 veranschaulicht. [Fen21]

Um einen Score für die Kalibrierungsqualität zu erhalten, können verschiedene, vom Calibration Plot abgeleitete Metriken verwendet werden. Häufig wird der Expected Calibration Error (ECE) genutzt [Fen21, GPSW17]. Er ist ein intervallbasiertes Maß für die durchschnittliche Abweichung der vorhergesagten von der tatsächlichen Certainty. Bezüglich des Calibration Plots ist der ECE die durchschnittliche Distanz zwischen der Kalibrierungskurve und der Diagonalen [Fen21]. Die T Intervalle sind für die Berechnung des ECE gleich breit und die Gewichtung des Fehlers pro Intervall wird von der Anzahl an Prädiktionen im Intervall $|B_t|$ bestimmt:

$$ECE = \sum_{t=1}^T \frac{|B_t|}{N} |\text{acc}(B_t) - \text{cert}(B_t)|. \quad (3.1)$$

N sei dabei die Gesamtzahl an Prädiktionen. Erwähnenswerte Metriken sind ebenfalls der Static Calibration Error (SCE), der die Prädiktionen für jede Klasse einzeln betrachtet [GTA⁺22], der Average Calibration Error (ACE), der den gleichgewichteten Fehler über alle Intervalle berechnet, und der Maximum Calibration Error (MCE) für den maximalen Fehler unter allen Intervallen [Fen21].

Kalibrierungsqualität des Neighborhood Uncertainty Classifiers

Von den Certainty Estimates des Neighborhood Uncertainty Classifiers (NUC) kann die Kalibrierung direkt evaluiert werden. Das externe Netzwerk ist bereits so trainiert, dass es die Certainty einer Prädiktion als Wahrscheinlichkeit zwischen 0 und 1 ausgibt. Abbildung 3.3 enthält Calibration Plots mit zugehörigen ECEs einer Anwendung der NUC-Methode für die Modelle CNN c10₁₀₀₀, CNN c10 und das EfficientNet. Die zugehörigen externen Certainty Netzwerke lernen anhand der Prädiktionen für die jeweiligen Trainingsdaten deren Korrektheit vorherzusagen und verwenden 10 nächstgelegene Nachbarn für die Certainty Estimation ($k = 10$).

Auffällig ist, dass viele Intervalle in den Calibration Plots leer sind und lediglich für das CNN c10 im mittleren Diagramm verläuft die Kalibrierungskurve annähernd auf der Diagonalen. Ein Faktor für die schlechte Performance des externen Netzwerks im linken und rechten Calibration Plot könnten das unzureichende Training aufgrund der im Vergleich zum gesamten Cifar10 Trainingsdatensatz kleinen Trainingsdatensätze sein. Anstatt 40000 liegen hier nur 1000 bzw. rund 6500 Trainingsbilder vor. Eine weitere Erklärung für die mangelnde Kalibrierungsqualität ist der große Unterschied zwischen Trainings- und Testfehler des CNN c10₁₀₀₀ und des EfficientNets. Das Certainty Netzwerk ist darauf trainiert, die Korrektheit der Prädiktionen des CNNs vorherzusagen. Somit unterscheidet es zwischen den Klassen korrekte und inkorrekte Prädiktion. Deep Neural Networks (DNNs), die auf einem kleinen Datensatz trainiert wurden, haben häufig eine deutlich höheren Fehler auf dem Test- als auf dem Trainingsdatensatz, da die Trainingsdaten unzureichend sind, um gut generalisieren zu können. Die hohe repräsentative Kapazität der Netzwerke ermöglicht so eine nahezu optimale Performance auf den Trainingsdaten. Das CNN c10₁₀₀₀ hat zum Beispiel auf den 1000 Trainingsbildern eine Accuracy von

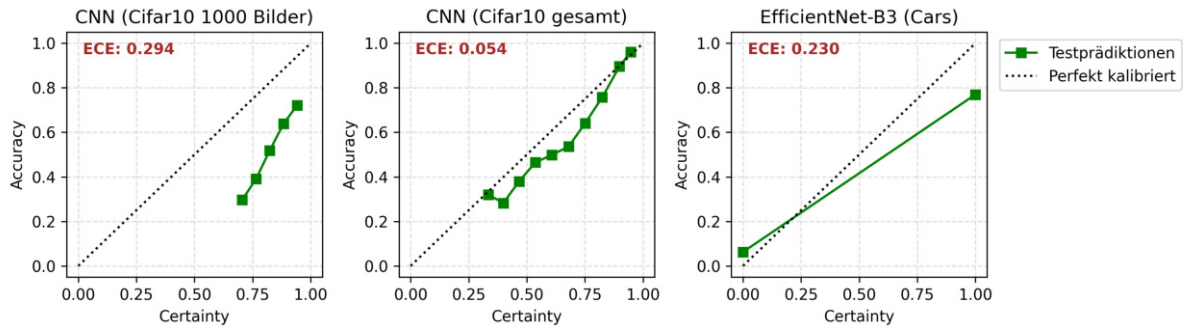


Abbildung 3.3: Calibration Plots und ECEs der NUC-Methode für CNNs, trainiert auf unterschiedlich vielen Bildern von Cifar10, und für das EfficientNet, trainiert auf dem Cars-Datensatz. Die Certainty Netzwerke wurde jeweils auf den Prädiktionen für diese Bilder trainiert und verwenden $k = 10$ Nachbarn. Für die Calibration Plots und die ECEs wurden die Testdatensätze und $T = 15$ Intervalle verwendet.

94% und auf dem Testdatensatz lediglich 54% (siehe Tabelle 3.1). Dies ist eine starke Domänenverschiebung in der Klassenverteilung des Certainty Netzwerks. Es lernt Prädiktionen häufiger als korrekt zu klassifizieren als es der realen Performance des CNNs entspricht. Die Certainty Scores der Testprädiktionen tendieren in diesem Fall dazu viel zu selbstsicher zu sein, was an der Kurve unterhalb der Diagonale im linken Calibration Plot erkennbar ist. Eine ähnliche Domänenverschiebung existiert für das EfficientNet. Die Accuracy auf Trainingsdaten beträgt fast 100%, weshalb das Certainty Netzwerk lernt, fast jede Prädiktion als sicher einzuschätzen. Dies sieht man gut im rechten Calibration Plot. Die Certainty Estimates sind schlecht über den Wertebereich von $(0, 1)$ verteilt und Prädiktionen mit einem Certainty Estimate von rund 1.0 haben eine durchschnittliche Accuracy von knapp unter 80%, was verglichen mit Tabelle 3.1 der Gesamtpformance des EfficientNets auf den Testdaten entspricht. Folglich befinden sich fast alle Prädiktionen im rechten Intervall des Calibration Plots und ihnen wird die viel zu hohe Certainty von annähernd 1.0 vorhergesagt.

Sind ausreichend viele Trainingsdaten vorhanden, so ist die Performance auf Trainings- und Testdatensatz bei einem gut generalisierenden Modell oft nicht derart verschieden. Das auf dem kompletten Cifar10 Datensatz trainierte CNN c10 hat eine Trainingsaccuracy von 88% und eine Testaccuracy von 74%. Die Certainty Estimates sind gut kalibriert und tendieren bei einem ECE von lediglich 0.05 nur sehr leicht dazu zu optimistisch zu sein, wie im mittleren Calibration Plot ersichtlich ist. Speicherplatzbedarf und Rechenaufwand wachsen jedoch mit der Anzahl der Bilder im Trainingsdatensatz, da die NUC-Methode für jedes Bild einen high-level Repräsentationsvektor speichert und für jede Testeingabe die Distanz zu allen gespeicherten Repräsentationsvektoren ermittelt. Das Certainty Netzwerk für das CNN c10 hat beispielsweise 40000 Repräsentationsvektoren gespeichert.

Im Folgenden wird versucht, die Domänenverschiebung beim Training des Certainty Netzwerks zu vermindern. Das externe Netzwerk wird nicht wie im Paper [RM19] darauf trainiert, die Performance eines CNNs für dessen Testdatensatz vorherzusagen, sondern für einen Teil des Validierungsdatensatzes. Die CNNs verwenden die Validierungsbilder im Trainingsprozess nur indirekt für die Reduzierung der Lernrate sowie für das Early Stopping. Folglich ist eine deutlich geringere Domänenverschiebung zwischen der Perfor-

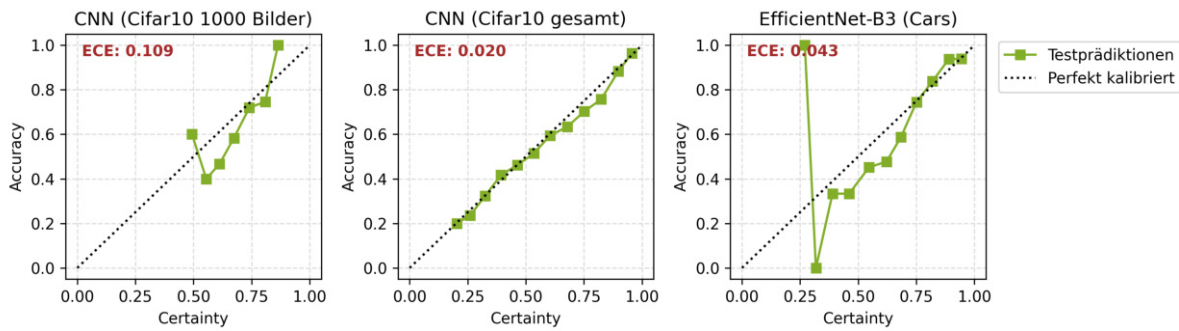


Abbildung 3.4: Calibration Plots und ECEs der NUC-Methode für CNNs trainiert auf unterschiedlich vielen Bildern von Cifar10 und für das EfficientNet trainiert auf dem Cars-Datensatz. Die Certainty Netzwerke lernen von Prädiktionen für den jeweiligen Validierungsdatensatz und verwenden $k = 10$ Nachbarn. Für die Calibration Plots und die ECEs wurden die Testdatensätze und $T = 15$ Intervalle verwendet.

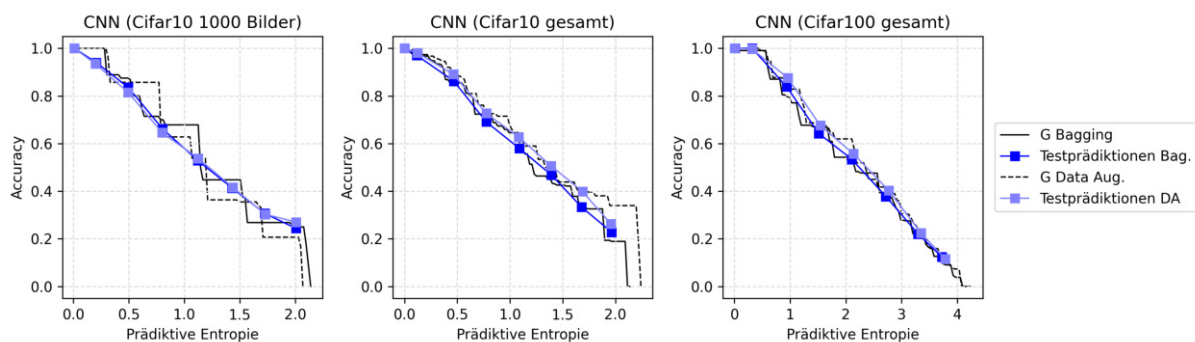
mance eines CNNs auf dem Validierungs- und Testdatensatz als zwischen Trainings- und Testdatensatz zu erwarten. Das Certainty Netzwerk lernt nun, die Korrektheit der Prädiktionen für die ersten 80% des Validierungsdatensatzes vorherzusagen. Für die eigentliche Validierung des Certainty Netzwerks werden die restlichen 20% verwendet. Die Ergebnisse sind in Abbildung 3.4 dargestellt.

Auffällig sind die deutlich geringeren Kalibrierungsfehler für das EfficientNet und das CNN c10₁₀₀₀. Einzelne starke Abweichungen von der Diagonalen, wie etwa im Fall des EfficientNets, sind nicht zwangsläufig ein Zeichen für eine schlechte Kalibrierung. In einem solchen Intervall sind möglicherweise nur sehr wenige Prädiktionen enthalten, die die gesamte Kalibrierung viel zu pessimistisch erscheinen lassen. Die Kalibrierungsqualität für das CNN c10 scheint sich ebenfalls verbessert zu haben. Dies ist jedoch für die hier betrachtete einmalige Ausführung der NUC-Methode kaum bewertbar. Beim Training des Certainty Netzwerks gibt es viele stochastische Faktoren, die dessen anschließende Performance beeinflussen können, weshalb am Ende dieses Abschnitts die ECEs mehrerer Certainty Netzwerke für ein Modell betrachtet werden. Aufgrund der von der Differenz zwischen Trainings- und Testfehler unabhängigen, guten Kalibrierung wird die soeben untersuchte Version der NUC-Methode für die weiteren Tests zusätzlich in Betracht gezogen.

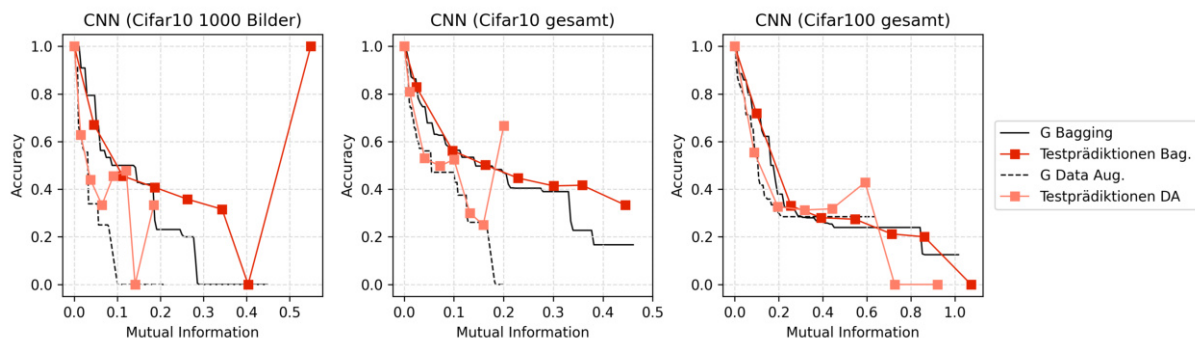
Kalibrierung der Sampling-basierten Methoden

Der Monte Carlo Dropout (MC Dropout) Ansatz und die Ensemble Methoden geben im Gegensatz zur NUC-Methode keinen Score zwischen 0 und 1 für die Certainty einer Prädiktion aus. Die hier verwendeten Metriken prädiktive Entropie (PE) und Mutual Information (MI) sind ein Maß für die prädiktive Unsicherheit mit Minimum bei 0 und unterschiedlichen Maximalwerten. Dies gilt auch für die Softmaxentropie eines Netzwerks, die, wie im Abschnitt 2.4 erwähnt, für Klassifizierungsprobleme zur Uncertainty Estimation verwendet werden kann.

Es gibt mehrere Techniken zur Kalibrierung solcher Uncertainty Estimates, die eine Übersetzung in Certainty Scores zwischen 0 und 1 ermöglichen. Ein Ansatz ist die isoto-



(a) Regressionsfunktion für die prädiktive Entropie (PE)



(b) Regressionsfunktion für die Mutual Information (MI)

Abbildung 3.5: Darstellung der isotonischen Regressionsfunktion G und der tatsächlichen durchschnittlichen Accuracy von Testprädiktionen für drei Bagging und Data Augmentation Ensembles. Die Regressionsfunktion wurde für Uncertainty Estimates (PE und MI) des Validierungsdatensatzes ermittelt. Ensemblemitglieder sind jeweils 5 CNNs für die Datensätze Cifar10 und Cifar100.

nische Regression. Mit diesem Regressionsverfahren wird eine ordnungserhaltende Abbildung erzeugt, die eine Reihe von Datenpunkten am besten annähert. Die von der Abbildung beschriebene Kurve hat als einzige Beschränkung, monoton steigend (oder fallend) zu sein, während die Distanz zu den Datenpunkten so gering wie möglich ist. [Fen21]

Mittels isotonischer Regression können die Uncertainty Estimates der Entropien und der Mutual Information auf kalibrierte Certainty Scores abgebildet werden. Je höher die Unsicherheit, desto geringer sollte der Certainty Score sein. Um diesen Zusammenhang zu erhalten, eignet sich eine monoton fallende isotonische Regression. Dies sei die Funktion $G : \mathbb{N}_0 \rightarrow [0, 1]$. G bildet das Uncertainty Estimate einer Prädiktion auf die approximierte Wahrscheinlichkeit für die Korrektheit der Prädiktion ab [Fen21]. Ein Datenpunkt zur Bestimmung der Regressionsfunktion besteht aus der vorhergesagten Unsicherheit $\text{uncert}(\hat{y})$ einer Prädiktion \hat{y} und deren Korrektheit $\text{acc}(\hat{y})$. Hierbei sei $\text{acc}(\hat{y}) = 1$ bei korrekter und 0 bei inkorrektur Prädiktion. Die Datenbasis wird den Prädiktionen des Modells für die Validierungsdaten des jeweiligen Datensatzes entnommen. Die Regressionsfunktion soll zur Testzeit des Modells auf neue, ungelabelte Testbeispiele angewandt werden. Daher ist es sinnvoll, zur Ermittlung der Funktion Daten zu verwenden, auf denen eine ähnliche Modellperformance wie für neue Daten zu erwarten ist. Somit ist der

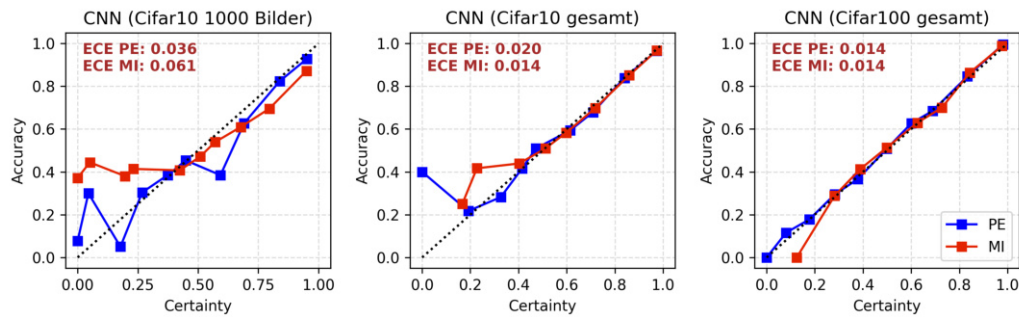


Abbildung 3.6: Calibration Plots und ECEs für drei Bagging Ensembles. Ensemblemitglieder sind jeweils 5 CNNs. Die Größe der Trainingsdatensätze ist über den Calibration Plots angegeben. Die Uncertainty Estimates der Shannon Entropie (SE) und Mutual Information (MI) wurden mit den Regressionsfunktionen in einen entsprechenden Wert für die Certainty zwischen 0 und 1 transformiert. Zur Evaluierung wurden der Cifar10 bzw. Cifar100 Testdatensatz und $T = 10$ Intervalle verwendet.

Validierungsdatensatz dem Trainingsdatensatz vorzuziehen. Der Testdatensatz wird zur Evaluierung verwendet.

Abbildung 3.5 zeigt exemplarische Regressionsfunktionen für die prädiktive Entropie und die Mutual Information der Ensemble Methoden Bagging und Data Augmentation. Die isotonische Regression wurde hier für drei Ensembles, die auf verschiedenen Datensätzen trainiert wurden, angewandt. Zusätzlich wurden Prädiktionen für Bilder des Testdatensatzes anhand ihrer Uncertainty Estimates in Intervalle gruppiert und deren Accuracy bestimmt. Dies dient der visuellen Evaluierung der Regressionsfunktion. Der erwartete Zusammenhang, dass Prädiktionen mit höherem Uncertainty Estimate öfter inkorrekt sind, zeigt sich in der monoton fallenden Accuracy der Testdaten bei steigender Unsicherheit. Es ist gut ersichtlich, dass sich die Accuracy der Prädiktionen für die Testdaten kaum von den Funktionswerten der Regressionsfunktion unterscheiden, vor allem für die Modelle CNN c10 und CNN c100. Jedoch ist nicht auszuschließen, dass eine Domänenverschiebung zwischen dem Validierungsdatensatz und den Bildern zur Testzeit die Übersetzung der Uncertainty Estimates mit der Regressionsfunktion verschlechtert. Auffällig ist, dass die Uncertainty Estimates für verschiedene Datensätze jeweils unterschiedlich skaliert sind. In Abbildung 3.5a erkennt man, dass eine Entropie von 2.0 für Cifar10 einer Accuracy von circa 0.2 und für Cifar100 einer Accuracy von über 0.5 entspricht. Da Cifar100 zwischen mehr Klassen als Cifar10 unterscheidet und somit der Ausgabevektor größer ist, sind hier generell höhere Entropien möglich, was dieses Verhalten erklärt. Die Regressionsfunktionen der Mutual Information hingegen verlaufen sogar für die beiden Ensemble Methoden unterschiedlich, wie Abbildung 3.5b aufzeigt. Da die Mutual Information ein Maß für die Varianz unter den Prädiktionen der Ensemblemitglieder ist, kann dies darauf hindeuten, dass sich die Mitglieder im Bagging Ensemble stärker unterscheiden als die im Data Augmentation Ensemble.

Die Variationen in den Ergebnissen für prädiktive Entropie und Mutual Information machen die Kalibrierung mit Techniken wie der isotonischen Regression für die Interpretation dieser Uncertainty Estimates dringend notwendig. Aus den unkalibrierten Werten kann nicht geschlossen werden, ob die Ausgabe sicher oder unsicher ist. Dieses Verhalten zeigt sich ebenfalls für das MC Dropout und die Softmaxentropien der CNNs.

Abbildung 3.6 zeigt Calibration Plots ausgewählter Bagging Ensembles nach Transformation der Uncertainty Estimates durch die jeweilige isotonische Regressionsfunktion. Die Kalibrierungskurven sind gut an die optimale Diagonale angenähert, was sich auch in den niedrigen ECEs widerspiegelt. Für das auf 1000 Bildern von Cifar10 trainierte Ensemble sind die ECEs ein wenig höher. Dies kann daran liegen, dass die Regressionsfunktion aufgrund des kleineren Validierungsdatensatzes nur unzureichend generalisieren kann, da sie lediglich anhand dieser Daten ermittelt wird.

Modelle, deren Calibration Plots bisher noch nicht betrachtet wurden, weisen keine neuen Informationen auf. Die Kalibrierungskurven mit den zugehörigen ECEs aller getesteten Methoden sind für jedes Modell separat in Anhang B. einsehbar. Da der Validierungsdatensatz des CNN $c10_{100}$ nur 25 Bilder umfasst, wird auf diesem kein Certainty Netzwerk trainiert.

Evaluierung der Kalibrierungsfehler

Fast jede untersuchte Methode führt nach mehrfacher Ausführung zu unterschiedlichen Ergebnissen. Die Trainingsprozesse der Ensemblemitglieder und des Certainty Netzwerks sind stochastisch. Das MC Dropout ist sogar zum Zeitpunkt der Anwendung stochastisch. Dies macht die mehrfache Ausführung der Methoden und die anschließende statistische Auswertung der Kalibrierungsfehler notwendig. Für die folgende Evaluierung wird jede Methode für jeden betrachteten Datensatz fünfmal angewandt. Die Certainty Netzwerke und Ensemblemitglieder werden jedes Mal neu trainiert.

In Tabelle 3.2 sind die Mittelwerte und Standardabweichungen für die verschiedenen ECEs zusammengefasst. Bis auf die Certainty Scores der NUC-Methoden sind alle Ausgaben mit isotonischer Regression kalibriert. Für die NUC-Methoden wurden $k = 10$ Nearest Neighbors verwendet. Es konnte kein Zusammenhang des Parameters k mit der Höhe des Kalibrierungsfehlers festgestellt werden, weshalb $k = 10$ zur Repräsentation der Kalibrierung dieser Methoden ausgewählt wurde.

Tabelle 3.2a zeigt, dass die Kalibrierungsfehler von Methoden, die isotonische Regression anwenden, für jedes Netzwerk geringer sind als die der NUC-Methoden. Dementsprechend könnte die Kalibrierung der NUC-Methoden durch Anwendung isotonischer Regression weiter verbessert werden, obwohl diese bereits einen Certainty-Score zwischen 0 und 1 vorhersagen. In dem Fall ist eine monoton steigende Regressionsfunktion von Nöten. Jedoch erhöht die zusätzliche isotonische Regression den Rechenaufwand der Methoden und die ECEs könnten sich bei einer Domänenveränderung zwischen Validierungs- und Testdaten erhöhen. Die ECEs bestätigen außerdem bereits getätigte Beobachtungen: die Kalibrierung des auf Validierungsdaten trainierten Certainty Netzwerks (NUC Va) ist besser als die des auf den Trainingsdaten trainierten (NUC Tr) und die Kalibrierungsfehler für die isotonische Regression fallen mit wachsender Größe des Validierungsdatensatzes. Letzteres lässt sich in Tabelle 3.2b ebenfalls für die Standardabweichungen feststellen².

²Bemerke, die Standardabweichungen der Softmaxentropie sind hier immer 0.0, da sich die Parameter des Modells bei jeder Wiederholung nicht verändern. Dies ist wichtig, um die Varianz von Methoden wie dem MC Dropout für die Anwendung auf einem festen Modell einschätzen zu können.

	CNN c10 ₁₀₀	CNN c10 ₁₀₀₀	CNN c10 ₁₀₀₀₀	CNN c10	CNN c100	Eff Cars
SE	0.092	0.041	0.026	0.016	0.012	0.018
MCD PE	0.083	0.076	0.022	0.012	0.015	0.019
MCD MI	0.157	0.082	0.027	0.010	0.012	0.015
Bag PE	0.135	0.045	0.020	0.013	0.018	0.018
Bag MI	0.212	0.049	0.018	0.010	0.016	0.016
DA PE	0.145	0.055	0.024	0.013	0.014	0.014
DA MI	0.234	0.049	0.021	0.013	0.015	0.020
NUC Tr	0.236	0.250	0.126	0.042	0.045	0.231
NUC Va	-	0.109	0.029	0.024	0.021	0.046

(a) Mittelwerte

	CNN c10 ₁₀₀	CNN c10 ₁₀₀₀	CNN c10 ₁₀₀₀₀	CNN c10	CNN c100	Eff Cars
SE	0.000	0.000	0.000	0.000	0.000	0.000
MCD PE	0.023	0.009	0.003	0.001	0.002	0.004
MCD MI	0.043	0.011	0.003	0.002	0.005	0.003
Bag PE	0.035	0.012	0.003	0.003	0.003	0.003
Bag MI	0.040	0.013	0.007	0.002	0.002	0.003
DA PE	0.034	0.009	0.004	0.004	0.003	0.003
DA MI	0.042	0.020	0.005	0.003	0.003	0.002
NUC Tr	0.127	0.087	0.034	0.017	0.031	0.001
NUC Va	-	0.017	0.002	0.004	0.002	0.002

(b) Standardabweichungen

Tabelle 3.2: Mittelwerte und Standardabweichungen der ECEs der Certainty Estimates jeder Methode von 5 Wiederholungen für alle getesteten Modelle. Die ECEs wurden mit $T = 15$ Intervallen berechnet und für die NUC-Methoden wurden $k = 10$ Nearest Neighbors verwendet. (Abkürzungen: PE - Prädiktive Entropie; MI - Mutual Information; SE - Softmaxentropie; MCD - Monte Carlo Dropout; Bag - Bagging Ensemble; DA - Data Augmentation Ensemble; NUC Tr - NUC trainiert auf Trainingsdaten; NUC Va - NUC trainiert auf Validierungsdaten)

Abschließend sei zu erwähnen, dass aus einem geringen Kalibrierungsfehler nicht geschlossen werden kann, dass die Uncertainty Estimates besonders aufschlussreich sind. Eine Uncertainty Estimation Methode hat für ein Modell, dessen Testaccuracy 80% beträgt, einen optimalen Kalibrierungsfehler von 0, wenn es unabhängig von einzelnen Prädiktionen stets eine Certainty von 0.8 vorhersagt. So eine Methode ist unbrauchbar. Im Idealfall sollen korrekte Prädiktionen anhand eines besonders geringen und inkorrekte Prädiktionen anhand eines hohen Uncertainty Estimates erkannt werden. Um zu überprüfen, ob die Uncertainty Estimates dieses Qualitätsmerkmal haben, sind weitere Tests notwendig.

Die in Tabelle 3.2 verwendeten Abkürzungen der Methoden werden ebenfalls für die Darstellung der Ergebnisse der folgenden Abschnitte genutzt.

3.3.2 Ergebnisse ausgewählter Evaluierungsmetriken

Prädiktionen eines neuronalen Netzwerks sind für Klassifizierungsaufgaben entweder korrekt oder inkorrekt. Bei der Uncertainty Estimation kann zusätzlich zwischen sicheren und unsicheren Prädiktionen unterschieden werden. Ähnlich wie bei der geläufigen Confusion Matrix können so quantitative Performance-Metriken für die Uncertainty Estimates definiert werden. [HKH⁺22, ASA⁺22]

Die übliche Confusion Matrix dient der Validierung eines binären Klassifizierers, indem korrekte und inkorrekte Prädiktionen für jede Klasse, *Positive* oder *Negative*, gezählt und in der Matrix dargestellt werden. Man unterscheidet *True Positive*, *False Positive*, *False Negative* und *True Negative*. [DG19]

Durch den Vergleich einer Prädiktion $\hat{\mathbf{y}}$ mit dem zugehörigen wahren Label \mathbf{y} wird festgestellt, ob $\hat{\mathbf{y}}$ zu den korrekten oder inkorrekten Ausgaben gruppiert wird. Die zugehörige Schätzung für die prädiktive Unsicherheit $uncert(\hat{\mathbf{y}})$ wird mit einem Schwellenwert verglichen und somit in die Gruppe sichere oder unsichere Prädiktion eingeteilt. Die Kombination der Gruppen resultiert in vier möglichen Unterteilungen, wie in Tabelle 3.3, der Uncertainty Confusion Matrix, zu sehen ist.

	Sicher	Unsicher
Korrekt	True Certain TC	False Uncertain FU
Inkorrekt	False Certain FC	True Uncertain TU

Tabelle 3.3: Uncertainty Confusion Matrix

Korrekte Prädiktionen, die anhand ihres Uncertainty Estimates als sicher eingestuft werden, gehören zur Kategorie *True Certain* (TC). Korrekte, aber unsichere Prädiktionen werden als *False Uncertain* (FU) bezeichnet. Die inkorrekten Ausgaben, die jedoch als sicher gelten, gehören zur Gruppe *False Certain* (FC) und die inkorrekten und unsicheren Prädiktionen zählen zu *True Uncertain* (TU). TU und TC sind die Idealfälle. Während FU situationsbasiert akzeptabel sein kann, sollten qualitative Uncertainty Estimates FC stets gering halten. [ASA⁺22, HKH⁺22]

Da die Vorhersage inkorrektur Prädiktionen durch hohe Schätzungen der prädiktiven Unsicherheit das Hauptziel der Uncertainty Estimation ist, sind unsichere Ausgaben von besonderem Interesse. Häufig verwendete quantitative Performance-Metriken im Bereich des maschinellen Lernens wie Precision und Recall fokussieren sich hauptsächlich auf die *True Positives* (TP), weshalb die unsicheren Prädiktionen als positive Gruppierung der gewöhnlichen Confusion Matrix betrachtet werden ($U \equiv P$). Entsprechend wurden die Metriken zur Evaluierung von Uncertainty Estimates folgendermaßen von M. Hasan et al. [HKH⁺22] und H. Asgharnezhad et al. [ASA⁺22] definiert:

Recall (U_{Rec}), auch Sensitivität oder True Positive Rate genannt, indiziert wie viele positive Ausgaben korrekt als positiv gelabelt wurden. Analog geht es bei der Uncertainty Estimation darum, wie viele der falschen Prädiktionen als unsicher klassifiziert werden. Die Metrik quantifiziert die Stärke einer Uncertainty Estimation Methode, die Unsicherheit für falsch klassifizierte Daten auszudrücken. Folglich ist U_{Rec} der Anteil

inkorrekt, unsicherer Prädiktion (TU) im Verhältnis zu allen inkorrekten Prädiktionen (TU+FC):

$$U_{Rec} = \frac{TU}{TU + FC} \quad (3.2)$$

Precision (U_{Pre}) ist bei der Uncertainty Estimation das Verhältnis von unsicheren und inkorrekten Prädiktionen (TU) zu allen unsicheren Prädiktionen (TU+FU). Ein hoher Wert deutet darauf hin, dass Prädiktionen, die von der Uncertainty Estimation Methode als unsicher eingestuft werden, tatsächlich inkorrekt sind:

$$U_{Pre} = \frac{TU}{TU + FU} \quad (3.3)$$

Specificity (U_{Spe}), auch als True Negative Rate bezeichnet, ist für die Uncertainty Estimation als der Anteil korrekter und sicherer Prädiktionen (TC) unter allen korrekten Prädiktionen definiert (TC+FU). Ein hoher Wert impliziert, dass die Uncertainty Estimation Methode die meisten korrekten Prädiktionen auch als korrekt vorhersagen kann:

$$U_{Spe} = \frac{TC}{TC + FU} \quad (3.4)$$

Für ideale Uncertainty Estimates haben diese drei Metriken jeweils den Wert 1.

Eine weitere, häufig verwendete Metrik ist die **False Positive Rate** (U_{FPR}). Sie steht in engem Zusammenhang mit der Specificity und hat den Wert 0 für ideale Schätzungen der prädiktiven Unsicherheit. Analog zur obigen Vorgehensweise nach [HKH⁺22] wird die Metrik für Uncertainty Estimates definiert als:

$$U_{FPR} = 1 - U_{Spe} = \frac{FU}{FU + TC} \quad (3.5)$$

Um Prädiktionen als sicher oder unsicher einzuteilen, muss ein Schwellenwert für die Grenze zwischen den beiden Gruppen definiert werden [HKH⁺22, ASA⁺22]. Dieser kann je nach Anwendung sehr unterschiedlich sein. Beispielsweise können Prädiktionen ab einer Certainty von 80% oder aber erst ab 95% als sicher gelten. Unterschiedliche Schwellenwerte resultieren in verschiedenen Werten der soeben aufgeführten Metriken, was die Evaluierung der Uncertainty Estimates verkompliziert.

Eine Lösung hierfür ist die AUROC Metrik. Sie bestimmt die Fläche unter der Receiver Operating Characteristic (ROC) Kurve und ermöglicht eine vom Schwellenwert unabhängige Evaluierung [HG18]. Die ROC-Kurve ist ein Graph, der Werte der False Positive Rate U_{FPR} auf die True Positive Rate (Recall) U_{Rec} abbildet. Sie zeigt auf, wie sich die True Positive Rate mit steigender False Positive Rate verhält. Für jeden möglichen Schwellenwert berechnet man hierfür beide Performancemetriken. Die False Positive Rate wird auf der x-Achse und die True Positive Rate auf der y-Achse abgetragen.

ROC-Kurven stellen in manchen Fällen die Qualität der Uncertainty Estimates zu optimistisch dar. Dies kann bei einer ungleichmäßigen Verteilung der Daten auf die Klassen eines Problems vorkommen. Eine Alternative für Probleme mit ungleichmäßig vielen Daten pro Klasse ist die Precision-Recall (PR) Kurve, welche für die Evaluierung der Methoden

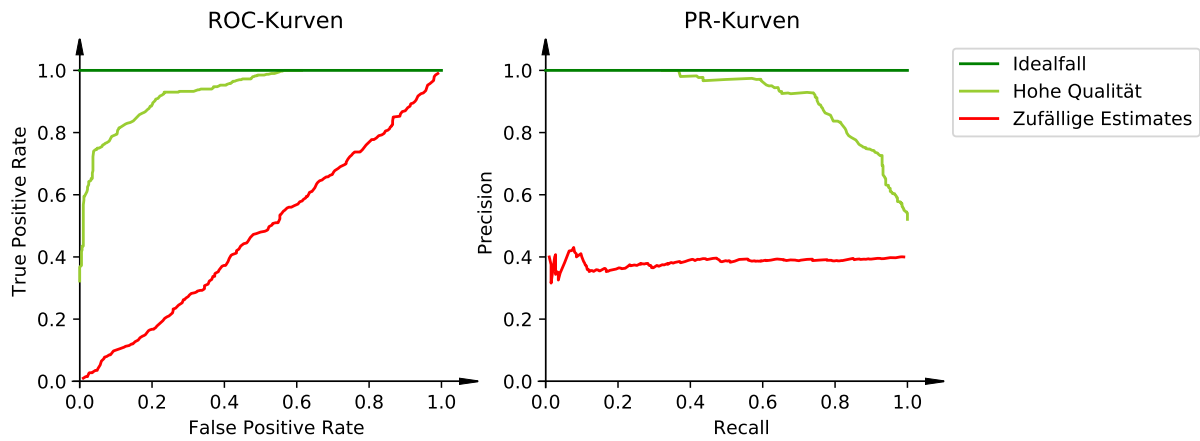


Abbildung 3.7: ROC- und PR-Kurven für Uncertainty Estimates verschiedener Qualität. Betrachtet werden 1000 Prädiktionen, die zu 60% korrekt sind.

neben der ROC-Kurve betrachtet wird. Hier werden Precision U_{Pre} und Recall U_{Rec} für sämtliche Schwellenwerte genauer betrachtet. [DG06, HG18]

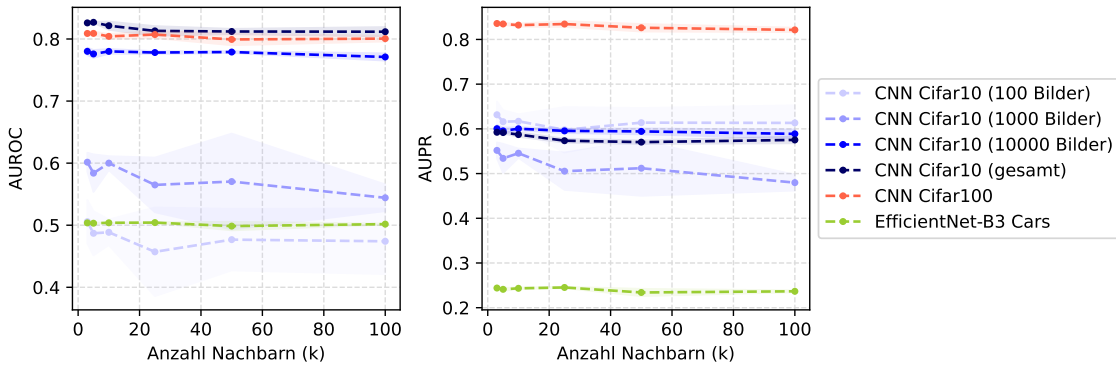
Abbildung 3.7 zeigt Beispiele für die ROC- und PR-Kurve von 1000 Prädiktionen, die zu 60% korrekt sind. Die ideale ROC-Kurve ist wie die ideale PR-Kurve die Gerade $y = 1$. Zufällige Uncertainty Estimates resultieren in einer Diagonalen als ROC-Kurve und ihre PR-Kurve verläuft in etwa waagerecht [FK15]. Der annähernd konstante Wert der PR-Kurve für zufällige Uncertainty Estimates entspricht der Precision [FK15], also dem Anteil der TUs unter allen unsicheren Prädiktionen. Da die unsicheren Prädiktionen zufällig gewählt werden, nähert sich die Precision dem Anteil der inkorrekten Prädiktionen unter allen Prädiktionen an. Für den in Abbildung 3.7 vorliegenden Fall ergibt dies eine Precision von 0.4.

Um konkrete Werte zu erhalten, die die Qualität der Uncertainty Estimates widerspiegeln, werden häufig die Flächen unter den Kurven verwendet. Perfekte Uncertainty Estimates haben sowohl unter der ROC-Kurve (AUROC) als auch unter der PR-Kurve (AUPR) eine Fläche von 1. Die AUROC von zufällig gewählten Uncertainty Estimates ist 0.5 und die AUPR in etwa der Precision selbst [HG18].

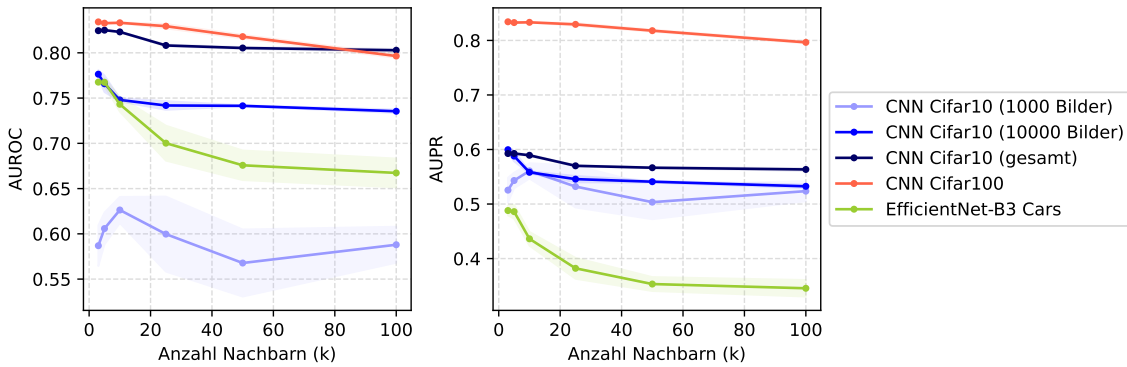
Wie im vorherigen Abschnitt 3.3.1 bei der Ermittlung von Kalibrierungsfehlern, sind zur Auswertung der Evaluierungsmetriken ebenfalls mehrere Ausführungen der Uncertainty Estimation Methoden notwendig. Verwendet werden fünf Wiederholungen jeder Methode. Hierbei erfolgen die Trainingsprozesse der Ensemblemitglieder und des externen Netzwerks vom Neighborhood Uncertainty Classifiers (NUC) jeweils neu. Zunächst werden AUROC und AUPR untersucht und anschließend die Metriken Precision, Specificity und Recall für ausgewählte Certainty-Schwellenwerte genauer betrachtet.

Kalibrierungsunabhängige Evaluierung

Da die ROC- und PR-Kurven nicht von bestimmten Schwellenwerten abhängig sind, können die unkalibrierten Uncertainty Estimates der Softmaxentropie, der prädiktiven Entropie und der Mutual Information verwendet werden. Dies hat den Vorteil, dass die direkte Ausgabe von Methoden wie dem Monte Carlo Dropout (MC Dropout) und den



(a) AUROCs und AUPRs für Certainty Netzwerke trainiert auf Trainingsdaten (NUC Tr)



(b) AUROCs und AUPRs für Certainty Netzwerke trainiert auf Validierungsdaten (NUC Va)

Abbildung 3.8: Performance der NUC-Methode angewandt auf alle untersuchten Modelle als Funktion der Anzahl Nachbarn k . Die AUROCs und AUPRs wurden für Prädiktionen der jeweiligen Testdaten bestimmt. Dargestellt sind die Mittelwerte nach fünffachem Training des Certainty Netzwerks und die Standardabweichungen (leicht schattiert).

Ensembles evaluiert wird. Die von der Methode unabhängige Qualität der Kalibrierungsfunktion hat keinen Einfluss auf das Ergebnis.

Als erstes werden verschiedene Werte des Parameters k der NUC-Methode untersucht. Dieser gibt an, wie viele nächstgelegene Nachbarn das externe Netzwerk zur Certainty Estimation verwendet. Die Nachbarn beziehen sich auf den high-level Repräsentationsraum eines Hauptmodells (siehe Kapitel 2.3.5). Für die Evaluierung wird zwischen 3, 5, 10, 25, 50 und 100 Nachbarn unterschieden und es werden zwei Versionen für das Training des Certainty Netzwerks verwendet. Das eine Netzwerk lernt, wie im Paper [RM19] vorgeschlagen, die Korrektheit der Prädiktionen des Hauptmodells für den Trainingsdatensatz vorherzusagen. Das zweite Netzwerk wird auf Prädiktionen für Bilder, die zur Validierung des Hauptmodells verwendet werden, trainiert. Diese Version der NUC-Methode ist motiviert durch die bessere Kalibrierung der Uncertainty Estimates (siehe Kapitel 3.3.1). Die AUROCs und AUPRs nach fünffacher Wiederholung der Trainingsprozesse sind in Abbildung 3.8 dargestellt.

Bei einigen Datensätzen verändern sich AUROC und AUPR kaum für verschiedene Werte des Parameters k , vor allem für die in Abbildung 3.8a verwendeten Certainty Netzwerke, die auf Trainingsdaten trainiert wurden. Jedoch ist oft ein leichter Fall der Metriken

mit steigendem k erkennbar, da Nachbarn von weiter entfernten Regionen betrachtet werden. Weniger als 10 Nachbarn scheinen ausreichend zu sein, um die beste Performance der NUC-Methode zu erhalten. Die Performance des auf den Validierungsdaten trainierten Certainty Netzwerks ist stärker vom Parameter k abhängig, vermutlich weil im Vergleich zu den größeren Trainingsdatensätzen weniger Repräsentationsvektoren pro Klasse gespeichert sind. Dadurch werden bereits bei kleinen k 's Nachbarn unterschiedlicher Klassen betrachtet. Für den kleinen Cars Datensatz verfügt der NUC Va lediglich über sechs Repräsentationsvektoren pro Klasse, da Cars sehr viele (196) Klassen hat. Bei einem k von größer als 6 werden stets Nachbarn betrachtet, die nicht mit der vom EfficientNet vorhergesagten Klasse übereinstimmen. Das Certainty Netzwerk hat dadurch Schwierigkeiten korrekte Prädiktionen als sicher einzuordnen, was den starken Fall des grünen Graphen in Abbildung 3.8b erklärt. Bezüglich der Standardabweichung ist zu vermerken, dass sich diese für beide Versionen des NUC mit der Größe des Datensatzes verringert. Auch mit wachsendem k ist teilweise ein Anstieg der Standardabweichung erkennbar. Für die weiteren Tests wird wegen benannter Gründe für alle Netzwerke $k = 3$ festgelegt.

Zum Vergleich aller Methoden sind in Tabelle 1 die Mittelwerte aller AUROCs und AUPRs für die jeweiligen Testdatensätze aufgelistet und die besten Werte markiert. Die Standardabweichungen der NUC-Methoden wurden bereits untersucht und die der restlichen Methoden sind gering und eher unauffällig. Sie können in Anhang A. eingesehen werden.

Die NUC-Methoden weisen, sobald ausreichend Daten für das Training des externen Netzwerks zur Verfügung stehen, die besten AUROCs und AUPRs auf. Dies ist für das CNN c10₁₀₀₀₀, das CNN c10 und das CNN c100 der Fall. Teilweise ist das auf den Trainingsdaten erlernte Certainty Netzwerk (NUC Tr) leicht besser als das auf Validierungsdaten trainierte (NUC Va). Auffällige Unterschiede zwischen den zwei Versionen existieren nur für das EfficientNet-B3 (Eff Cars). AUROC und AUPR der Methode NUC Tr sind hier deutlich schlechter und entsprechen denen von zufälligen Certainty Estimates. Die Trainingsaccuracy des EfficientNets für den Cars Datensatz beträgt fast 100% (vergleiche Tabelle 3.1), sodass das Certainty Netzwerk keine inkorrekten Prädiktionen für das eigene Training hat, was die schlechte Performance erklärt. Dies hat sich bereits im vorherigen Abschnitt negativ auf die Kalibrierung ausgewirkt. Nicht nur für das EfficientNet, sondern auch für das CNN c10₁₀₀ entsprechen die AUROC und die AUPR denen von zufälligen Estimates, da auch hier bei einer Trainingsaccuracy von 93.0% viel zu wenig inkorrekte Prädiktionen (nur sieben) für das Training des externen Netzwerks vorliegen. Weil der Validierungsdatensatz für das CNN c10₁₀₀ nur 25 Bilder umfasst, ist das alternative Training des Certainty Netzwerks auf diesen Bildern (NUC Va) ebenfalls keine Option.

Für die Netzwerke CNN c10₁₀₀, CNN c10₁₀₀₀ und das EfficientNet, bei denen nur wenige Daten für das Modelltraining vorhanden sind, weisen die Entropien verschiedener Uncertainty Estimation Methoden die besten AUROCs und AUPRs auf. Tabelle 3.4a verdeutlicht, dass die prädiktive Entropie des mit Data Augmentation trainierten Ensembles (DA PE) häufig mit die größte AUROC hat. Bezüglich der AUPRs ist die prädiktive Entropie des Bagging Ensembles (Bag PE) besser. Die Ergebnisse der prädiktiven Entropie des MC Dropouts (MCD PE) und der einfachen Softmaxentropie des Netzwerks (SE) verhalten sich ähnlich zu denen der Ensembles.

	CNN c10 ₁₀₀	CNN c10 ₁₀₀₀	CNN c10 ₁₀₀₀₀	CNN c10	CNN c100	Eff Cars
SE	0.642	0.728	0.780	0.822	0.785	0.839
MCD PE	0.650	0.721	0.773	0.805	0.773	0.864
MCD MI	0.606	0.683	0.735	0.762	0.682	0.862
Bag PE	0.640	0.718	0.771	0.804	0.775	0.836
Bag MI	0.522	0.640	0.712	0.758	0.698	0.826
DA PE	0.649	0.727	0.774	0.809	0.779	0.861
DA MI	0.537	0.645	0.697	0.750	0.692	0.846
NUC Tr	0.506	0.602	0.780	0.826	0.809	0.504
NUC Va	-	0.587	0.776	0.824	0.804	0.768

(a) AUROCs

	CNN c10 ₁₀₀	CNN c10 ₁₀₀₀	CNN c10 ₁₀₀₀₀	CNN c10	CNN c100	Eff Cars
SE	0.734	0.658	0.591	0.583	0.818	0.639
MCD PE	0.732	0.655	0.584	0.552	0.799	0.650
MCD MI	0.695	0.607	0.526	0.473	0.709	0.648
Bag PE	0.747	0.656	0.608	0.563	0.796	0.670
Bag MI	0.664	0.567	0.522	0.475	0.713	0.646
DA PE	0.729	0.648	0.581	0.559	0.785	0.616
DA MI	0.641	0.558	0.487	0.453	0.686	0.581
NUC Tr	0.632	0.552	0.600	0.592	0.836	0.244
NUC Va	-	0.525	0.600	0.593	0.834	0.488

(b) AUPRs

Tabelle 3.4: AUROCs und AUPRs aller Methoden zur Uncertainty Estimation. Es erfolgten fünf Wiederholungen, von denen die Mittelwerte gebildet wurden. Für die NUC-Methoden wurden $k = 3$ Nearest Neighbors gewählt. Der beste Wert sowie Werte, die um maximal 0.005 vom besten Ergebnis abweichen, sind optisch hervorgehoben.

Die gute Performance der Softmaxentropie (SE) erscheint aufgrund der Einfachheit dieser Methode überraschend. Obwohl die prädiktive Entropie die aleatorische und die epistemische Unsicherheit erfassen und somit die gesamte prädiktive Unsicherheit messen kann, sind die AUROCs und AUPRs der Softmaxentropie ähnlich gut und teilweise sogar besser. Die Softmaxentropie erfasst hauptsächlich die aleatorische Unsicherheit und wird von der epistemischen nur teilweise beeinflusst [GTA⁺22]. Die guten Ergebnisse zeigen jedoch, dass für Klassifizierungsprobleme nicht zwangsläufig auf komplizierte Uncertainty Estimation Techniken zurückgegriffen werden muss, um die prädiktive Unsicherheit abschätzen zu können. Nachteilig an der Verwendung der einzelnen Softmaxausgabe bleibt, dass die epistemische Unsicherheit nicht von der aleatorischen unterschieden werden kann. Dies ist bei Methoden wie dem MC Dropout und den Ensembles zum Beispiel durch die zusätzliche Betrachtung der Mutual Information möglich.

Vergleicht man die Mutual Information vom MC Dropout Ansatz oder den Ensemble Methoden mit der zugehörigen prädiktiven Entropie, fällt auf, dass diese für fast jedes Netz-

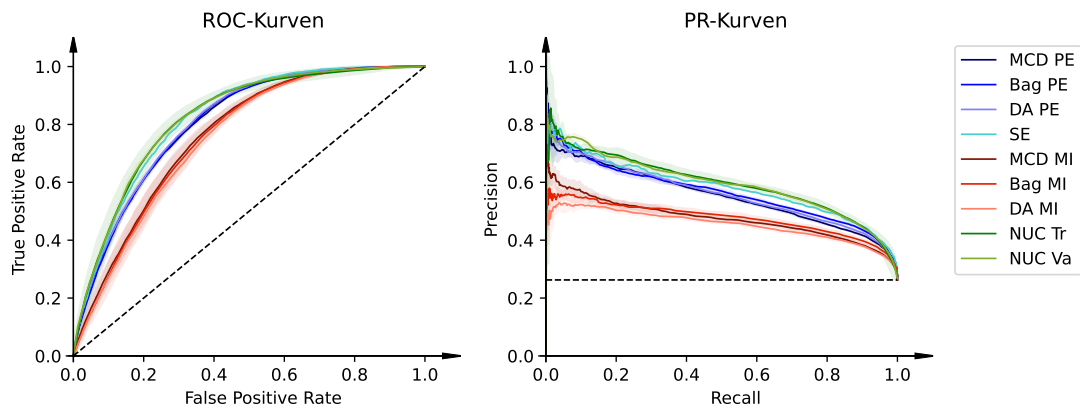


Abbildung 3.9: ROC- und PR-Kurven für Uncertainty Estimates aller betrachteten Methoden, angewandt auf das CNN c10. Die Unsicherheit wurde für Prädiktionen des CNNs für die Bilder des Cifar10 Testdatensatzes ermittelt. Dargestellt sind die Mittelwerte nach fünffacher Anwendung der Methoden und die Standardabweichungen (leicht schattiert). Kurven zufälliger Estimates sind gestrichelt gekennzeichnet. Zur besseren Lesbarkeit sind Uncertainty Estimates, die aus ähnlichen Methoden oder Metriken resultieren, vom selben Farbton.

werk kleinere AUROCs und AUPRs aufweist. Abbildung 3.9 verdeutlicht dies am Beispiel des Modells CNN c10. Hier sind die ROC- und PR-Kurven für jede Methode visualisiert. Die Kurve jeder Mutual Information verläuft deutlich unter der anderer Metriken. Dies kann daran liegen, dass die Mutual Information lediglich die epistemische Unsicherheit und folglich nicht die gesamte prädiktive Unsicherheit erfasst.

Kalibrierungsabhängige Evaluierung

Performance-Metriken wie die AUROC und die AUPR ermöglichen eine schwellenwertunabhängige Evaluierung der Uncertainty Estimates, indem die Metriken Precision, False Positive Rate und Recall für jeden möglichen Schwellenwert im Wertebereich der Estimates ausgewertet werden. Jedoch sind in der Anwendung der Certainty Estimation bestimmte Schwellenwerte wichtiger als andere. Sinnvoll sind Werte zwischen 0 und 1, die angeben, ab wann eine Prädiktion als sicher gilt. Als interessant erweisen sich vor allem hohe Werte, da für die wenigsten Anwendungen eine Certainty von beispielsweise 0.5 (50%) ausreicht, um der Prädiktion zu vertrauen. Demnach wird nun die Performance der Uncertainty Estimation Methoden nochmals genauer für Schwellenwerte ab 0.6 untersucht. Diese Schwellenwerte, die als Wahrscheinlichkeit für die Korrektheit der Prädiktionen interpretiert werden, erfordern die Verwendung kalibrierter Certainty Estimates. Wie im Abschnitt 3.3.1 beschrieben, wird isotonische Regression für die Kalibrierung der Ausgaben von Softmaxentropie (SE), prädiktiver Entropie (PE) und Mutual Information (MI) genutzt. Die Certainty Estimates der NUC-Methoden haben bereits einen Wertebereich von (0, 1), sodass sie direkt evaluiert werden können.

Im vorherigen Abschnitt 3.3.1 wurde festgestellt, dass die Kalibrierungsfehler steigen, je weniger Daten für das Training der Modelle verwendbar sind. Daher wird die kalibrierungsabhängige Evaluierung nicht für die Modelle und Ensembles, die lediglich auf Teilen

des Cifar10 Datensatzes trainiert wurden, ausgeführt. Im Speziellen wird sich auf das CNN c100, das CNN c10 und das EfficientNet fokussiert.

Die Metriken Precision, Specificity und Recall werden getrennt voneinander als Funktionen der Schwellenwerte betrachtet, um die Unterschiede in der Performance der Methoden zu verdeutlichen. Zu erwarten ist, dass die Methoden, deren AUROC und AUPR am größten sind, auch die höchsten Ergebnisse für die Metriken Precision, Specificity und Recall erzielen. Optimal ist für alle drei Metriken der Wert 1.

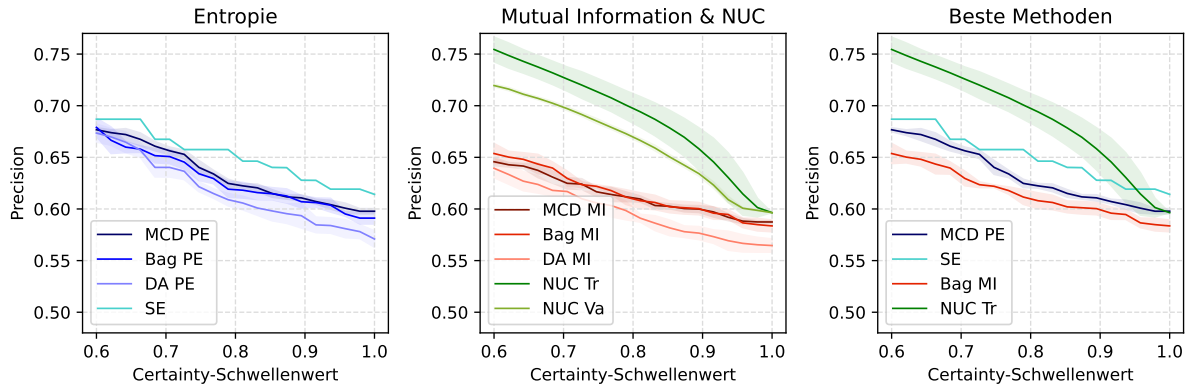
Abbildung 3.10 zeigt die Resultate für das CNN c100 und die entsprechenden Ensembles auf dem Cifar100 Testdatensatz. Es ist deutlich erkennbar, dass es keine Methode gibt, die für alle drei Metriken das beste Ergebnis liefert. Ein solches Verhalten ist plausibel, da gute Ergebnisse für alle drei Metriken teilweise in Konkurrenz zueinander stehen. Beispielsweise haben Certainty Estimates, die Ausgaben tendenziell als zu sicher einschätzen, nur wenig Prädiktion der Kategorie *False Uncertain* (FU) und viele der Kategorie *False Certain* (FC). Dies ergibt zwar eine hohen Specificity $U_{Spe} = \frac{TC}{TC+FU}$, aber gleichzeitig auch einen geringen Recall $U_{Rec} = \frac{TU}{TU+FC}$, was bei der NUC-Methode mit dem auf Trainingsdaten trainierten Certainty Netzwerk (NUC Tr) zu beobachten ist (dunkelgrüne Graphen). Bereits im Abschnitt 3.3.1 wurde festgestellt, dass die Certainty Estimates dieser Version der NUC-Methode zu selbstsicher sind.

Auffällig ist auch die hohe Distanz, mit der die Graphen der NUC-Methoden für die Precision und die Specificity über den anderen Graphen verlaufen. Das kann ein Grund für die hohen AUROCs und AUPRs der NUCs für das CNN c100 sein. Aus der hohen Precision ist zu schlussfolgern, dass Prädiktionen, die von den NUC-Methoden als unsicher vorhergesagt werden, häufiger tatsächlich inkorrekt sind. Die hohe Specificity deutet darauf hin, dass korrekte Prädiktionen öfter als sicher eingeschätzt werden als von anderen Methoden. Ein Nachteil von NUC Tr ist jedoch, dass die Standardabweichung hier größer ist als bei den anderen Methoden.

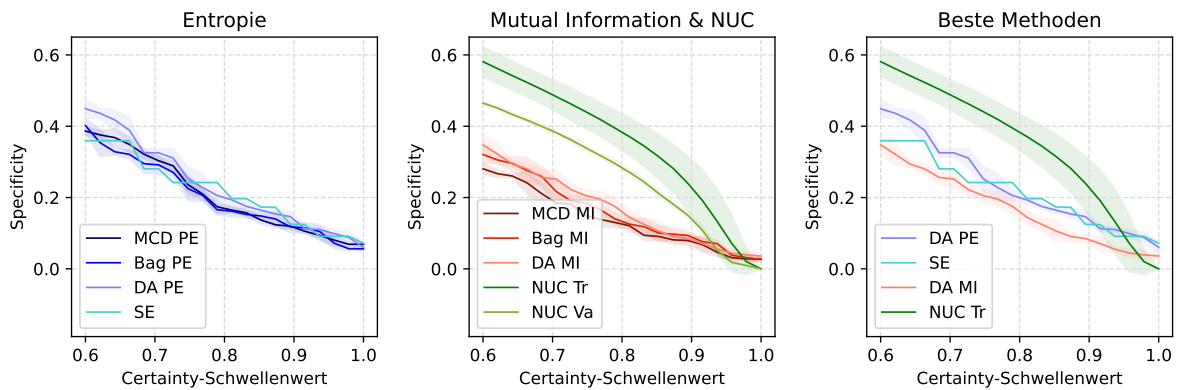
Beim genaueren Betrachten der Graphen in den rechten Diagrammen fällt auf, dass Certainty Estimates der prädiktiven Entropie tendenziell eine höhere Precision und Specificity als die der Mutual Information haben. Anhand der großen AUROCs und AUPRs der Entropien in Tabelle 1 ist die bessere Performance zu erwarten. Lediglich beim Recall ist die Mutual Information einer Methode etwas besser als die zugehörige prädiktive Entropie, was bedeutet, dass die Mutual Information inkorrekte Prädiktionen häufiger als unsicher erkennt. Die überraschend gute Performance der Softmaxentropie (SE) bezüglich der Metriken AUROC und AUPR zeigt sich auch in den Graphen für die Metriken Precision, Specificity und Recall. In den Graphiken auf der linken Seite von Abbildung 3.10 wird deutlich, dass die Softmaxentropie für jede der drei Metriken zu den besten Entropien gehört.

Beim Ranking der Performance der einzelnen Methoden sind vereinzelt Unterschiede in Abhängigkeit vom Schwellenwert bemerkbar. Zum Beispiel ist in Abbildung 3.10b die Specificity der Softmaxentropie (SE) für den Schwellenwert 0.7 die schlechteste und für 0.8 hingegen die beste Entropie. Derartige Unterschiede werden bei der Evaluierung mit AUROC und AUPR nicht sichtbar.

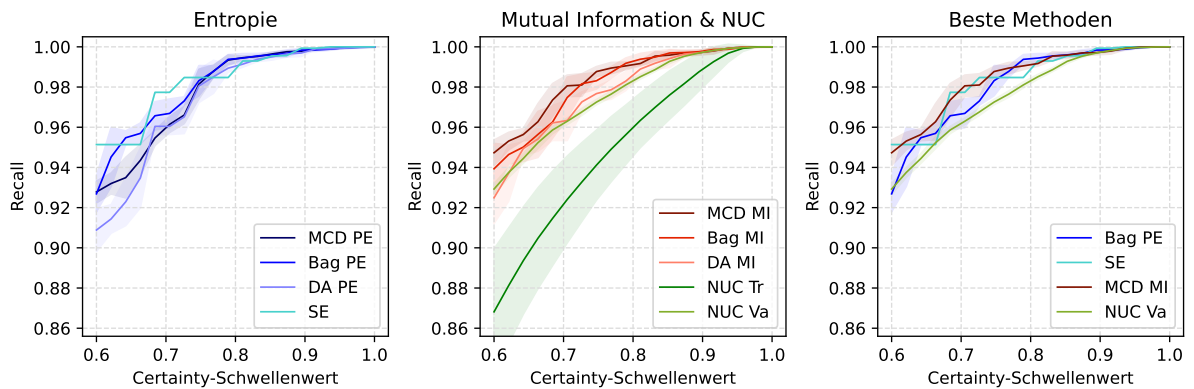
Die Ergebnisse der Methoden angewandt auf das CNN für den Cifar10 Datensatz (CNN c10) ähneln den in Abbildung 3.10 für das CNN c100 dargestellten Resultaten. Die genauen Funktionen für die Precision, die Specificity und den Recall für das CNN c10



(a) Precision

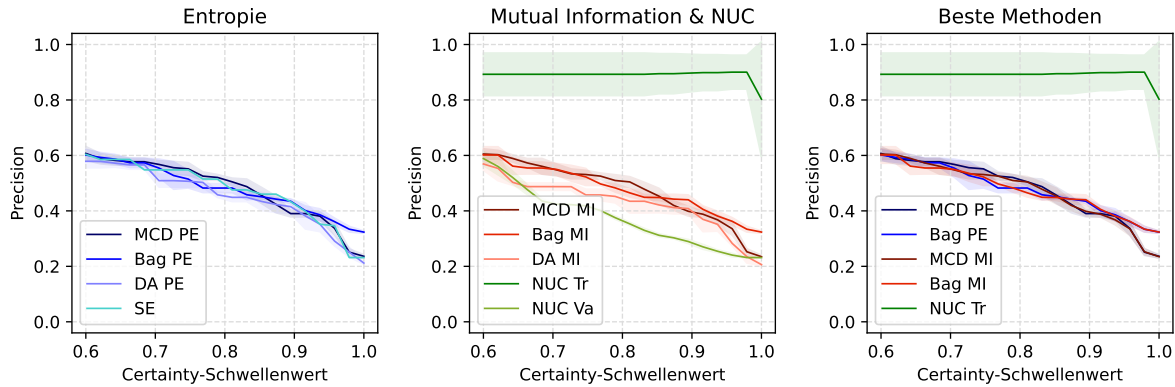


(b) Specificity

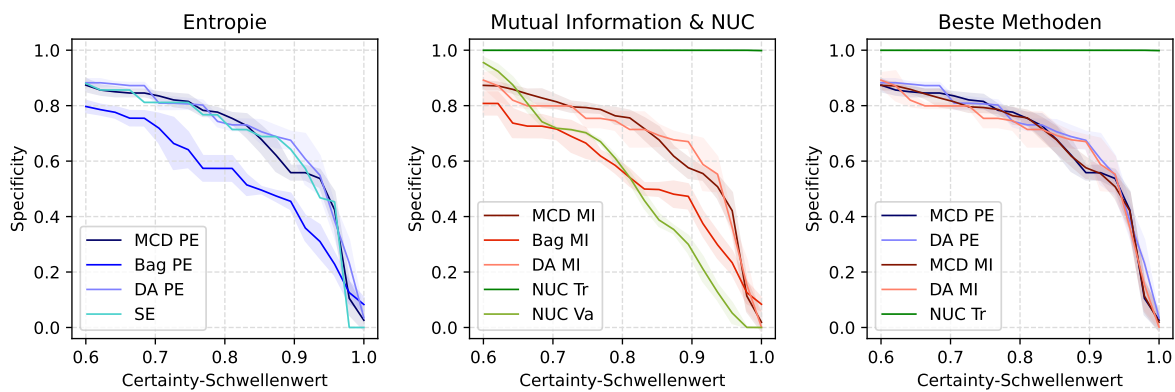


(c) Recall

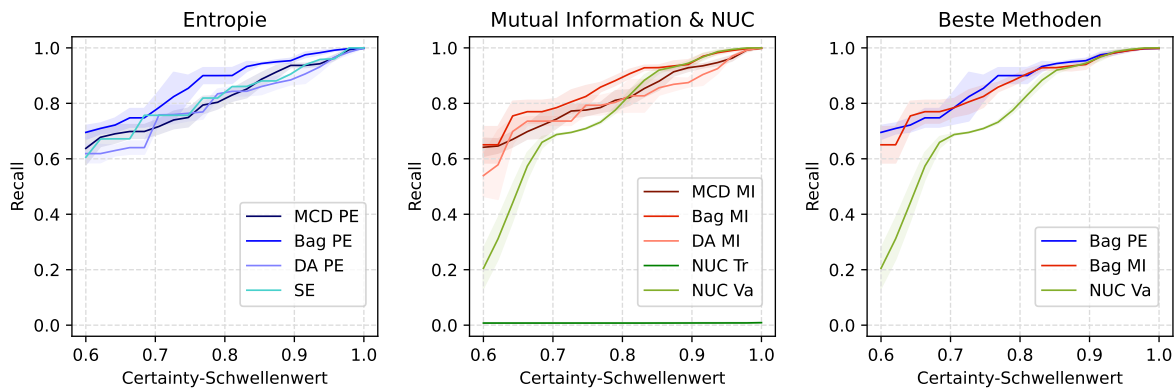
Abbildung 3.10: Darstellung der Metriken Precision, Specificity und Recall für die Uncertainty Estimation Methoden als Funktion von Certainty-Schwellenwerten für das CNN c100. Funktionswerte sind die Mittelwerte der Metriken nach fünffacher Ausführung der Methoden. Standardabweichungen sind leicht schattiert visualisiert. Zur besseren Übersichtlichkeit werden Uncertainty Estimates einzelner Metriken zunächst getrennt voneinander betrachtet und anschließend werden die besten Ergebnisse in einer separaten Graphik erneut dargestellt.



(a) Precision



(b) Specificity



(c) Recall

Abbildung 3.11: Ergebnisse der Metriken Precision, Specificity und Recall für die Uncertainty Estimation Methoden angewandt auf das EfficientNet-B3. Dargestellt sind die Mittelwerte der Metriken nach fünffacher Ausführung der Methoden sowie die Standardabweichungen (leicht schattiert). Zur besseren Übersichtlichkeit werden Uncertainty Estimates einzelner Metriken zunächst getrennt voneinander betrachtet und anschließend werden die besten Ergebnisse in einer separaten Graphik erneut dargestellt.

sind in Anhang D. einsehbar. Abschließend wird noch kurz auf die Graphen für das EfficientNet in Abbildung 3.11 eingegangen. Wie beim CNN c100 festgestellt, unterscheidet sich der Kurvenverlauf der NUC-Methoden, vor allem von NUC Tr, deutlich von dem der anderen Methoden. Ein Unterschied im Vergleich zu Abbildung 3.10 ist das ähnliche Verhalten von prädiktiver Entropie und Mutual Information derselben Uncertainty Estimation Methode. Beispielsweise verlaufen die Kurven der prädiktiven Entropie und Mutual Information des MC Dropout Ansatzes in Abbildung 3.11b fast übereinander.

Je nach Problemstellung kann man sich bei der Entscheidung, welche Methode verwendet werden soll, auf einzelne Metriken fokussieren. Häufig sind Prädiktionen der Klasse *False Certain* FC besonders schwerwiegend. In diesem Fall ist ein hoher Recall besonders wichtig. Jedoch ist davon abzuraten, nur eine der drei Metriken auszuwerten, denn das kann zu Fehlschlüssen führen, was gut am Beispiel der Graphen in Abbildung 3.11 erklärbar ist. Das auf den Trainingsdaten trainierte Certainty Netzwerk (NUC Tr) hat hier für den Cars Trainingsdatensatz gelernt, fast jeder Prädiktion eine Certainty von nahezu 1.0 zuzuweisen, denn das EfficientNet hat auf diesem eine Accuracy von 0.999. Somit gibt es für beliebige Schwellenwerte kaum unsichere Prädiktionen, die Menge FU ist fast leer und die Specificity $U_{Spe} = \frac{TC}{TC+FU}$ nahezu optimal. Auch die Precision ist deutlich besser als die der anderen Methoden. Jedoch gilt $FC \gg TU$, weshalb der Recall $U_{Rec} = \frac{TU}{TU+FC}$ fast bei 0 liegt. Optimale Werte bei einer Metrik sind folglich nicht zwangsläufig ein gutes Zeichen.

Zusammenfassend ermöglicht die kalibrierungsabhängige Evaluierung eine detailliertere Untersuchung der Performance der Metriken und vor allem eine Auswertung für einzelne, relevante Schwellenwerte. Nachteile sind, dass die Kalibrierungsqualität und somit auch die Qualität von Kalibrierungstechniken, wie der isotonischen Regression, einen Einfluss auf die Resultate haben. Ist wie beim aktiven Lernen nur die Ordnung von Prädiktionen nach ihrer Unsicherheit interessant sind die Metriken AUROC und AUPR zu bevorzugen. Bei solchen Anwendungen sind Schwellenwerte vernachlässigbar.

3.3.3 Out-of-Domain Erkennung

Eine weitere Anwendung der Uncertainty Estimation ist die Out-of-Domain Erkennung. Anhand einer hohen prädiktiven Unsicherheit kann erkannt werden, ob ein Sample nicht aus der Ziel-Domäne des Netzwerks stammt.

Um die Methoden zur Uncertainty Estimation auf Out-of-Domain (OOD) Daten zu testen, wird jeweils ein weiteres Netzwerk auf den Datensätzen Cifar10, Cifar100 und Cars trainiert, wobei für bestimmte Klassen alle Trainingsbilder entfernt werden. Diese Klassen sind für die Netzwerke unbekannt und Bilder dieser Klassen gehören nicht zu ihren Ziel-Domänen.

Prädiktionen für OOD Daten haben eine hohe epistemische Unsicherheit, da das Modell über kein OOD Wissen verfügt und somit deren Verteilung nicht kennt [GTA⁺22]. Die Mutual Information und prädiktive Entropie von Methoden wie den Ensembles und des MC Dropouts können die epistemische Unsicherheit erfassen, während die Softmaxentropie der einfachen Netzwerkausgabe etwas weniger von der epistemischen Unsicherheit beeinflusst wird, wie das Anwendungsbeispiel in Abschnitt 2.4 gezeigt hat. Für die Softmaxentropie

ist daher bei diesem OOD Test eine schlechtere Performance anzunehmen. T. Ramalho et al. [RM19] zeigten, dass der Neighborhood Uncertainty Classifier besonders für die Erkennung von OOD Daten geeignet ist, weshalb für die NUC-Methoden gute Resultate erwartet werden.

Damit stets genug OOD Daten zur Evaluierung vorliegen, werden die Netzwerke lediglich auf der Hälfte der ursprünglichen Klassen der Datensätze trainiert. Die Ausgabeschichten haben, dieser Domäne entsprechend, nur halb so viele Neuronen. Anschließend werden Prädiktionen für die OOD Daten bestimmt und deren Unsicherheit abgeschätzt. Für jede Prädiktion ist eine hohe Unsicherheit zu erwarten, da alle Prädiktionen inkorrekt sind. Für den OOD Test ist es jedoch wichtig, nicht nur die Prädiktionen für die OOD Bilder zu betrachten. Sonst erreicht eine Uncertainty Estimation Methode, die generell für jede Prädiktion, ob korrekt oder inkorrekt, eine hohe Unsicherheit vorhersagt, perfekte Ergebnisse. Vielmehr sollen die OOD Daten unter Daten aus der bekannten Domäne anhand eines höheren Uncertainty Estimates erkannt werden. Daher werden der Menge an OOD Daten Testbilder bekannter Klassen hinzugefügt. Von den Prädiktionen der Netzwerke für diese Bilder schätzen die verschiedenen Uncertainty Estimation Methoden die prädiktive Unsicherheit ab. Anschließend wird die Performance der Uncertainty Estimates wie im vorherigen Abschnitt 3.3.2 anhand der Metriken AUROC und AUPR evaluiert. Dieses Vorgehen wurde ebenfalls von [RM19] für die Bewertung der OOD Erkennung verwendet.

Neben den bereits betrachteten Uncertainty Estimation Methoden, ist das Maximum des Softmaxvektors der Modellausgabe (Max Soft) für den OOD Test interessant. Wie in Kapitel 2.2.4 erläutert, tendiert der Softmaxscore der vorhergesagten Klasse eines gut trainierten Modells dazu, viel zu selbstsicher zu sein. Dies ist der größte Wert im Softmaxvektor. Für die Erkennung von OOD Daten ist somit eine schlechte Performance zu erwarten, wie im selbigen Kapitel beschrieben. Anhand der Performance des maximalen Softmaxscores kann eingeschätzt werden, wie groß die Verbesserung der OOD Erkennung durch bewährte Methoden der Uncertainty Estimation ist.

Die Ergebnisse sind in Tabelle 3.5 zusammengefasst. Für den Cifar10 Datensatz zeigt sich wie erwartet, dass alle Methoden höhere AUROCs und AUPRs und somit eine bessere Qualität aufweisen als der maximale Softmaxscore. Bei Cifar100 hingegen ist die Performance vieler Methoden schlechter als die des maximalen Softmaxscores. Dies kann damit zusammenhängen, dass das CNN für die Trainingsdaten von Cifar100 nur eine Accuracy von 62% erreicht. Die Prädiktionen sind oft inkorrekt, was die Softmaxscores klein hält, da die vielen inkorrekten Prädiktionen andernfalls beim Training einen sehr hohen Loss zur Folge hätten. Somit sind auch die maximalen Softmaxscores für die OOD Daten klein und nicht zu selbstsicher.

Abgesehen von den NUC-Methoden zeigen wieder alle Uncertainty Estimates für den Cars Datensatz eine bessere Performance bei der OOD Erkennung als der maximale Softmaxscore. Die schlechte Performance der NUCs ist hier analog zu den vorigen Tests damit zu begründen, dass NUC Tr aufgrund der hohen Trainingsaccuracy des EfficientNets keine inkorrekten Trainingsbeispiele hat und für das Training von NUC Va zu wenig Validierungsdaten zur Verfügung stehen. Während die Standardabweichungen in Tabelle 3.5b für alle Methoden sehr klein sind, fallen die hohen Standardabweichungen der AUROC und AUPR von NUC Tr negativ auf, was mit den unzureichenden inkorrekten Trainingsbeispielen zusammenhängen kann.

	Cifar10 (CNN) AUROC / AUPR	Cifar100 (CNN) AUROC / AUPR	Cars (EfficientNet) AUROC / AUPR
MCD PE	0.784 / 0.961	0.736 / 0.970	0.815 / 0.941
MCD MI	0.777 / 0.959	0.710 / 0.965	0.812 / 0.939
Bag PE	0.759 / 0.956	0.737 / 0.969	0.815 / 0.939
Bag MI	0.727 / 0.948	0.699 / 0.961	0.788 / 0.927
DA PE	0.776 / 0.958	0.746 / 0.968	0.861 / 0.945
DA MI	0.762 / 0.955	0.710 / 0.960	0.841 / 0.935
NUC Tr	0.727 / 0.946	0.755 / 0.973	0.490 / 0.794
NUC Va	0.730 / 0.947	0.752 / 0.973	0.692 / 0.892
SE	0.746 / 0.953	0.744 / 0.972	0.783 / 0.929
Max Soft	0.726 / 0.947	0.745 / 0.972	0.767 / 0.922

(a) Mittelwerte

	Cifar10 (CNN) AUROC / AUPR	Cifar100 (CNN) AUROC / AUPR	Cars (EfficientNet) AUROC / AUPR
MCD PE	0.001 / 0.000	0.001 / 0.000	0.002 / 0.001
MCD MI	0.001 / 0.000	0.002 / 0.000	0.002 / 0.001
Bag PE	0.005 / 0.001	0.005 / 0.001	0.003 / 0.003
Bag MI	0.004 / 0.001	0.008 / 0.002	0.003 / 0.003
DA PE	0.002 / 0.000	0.003 / 0.001	0.003 / 0.001
DA MI	0.004 / 0.001	0.005 / 0.001	0.004 / 0.002
NUC Tr	0.007 / 0.002	0.003 / 0.001	0.077 / 0.036
NUC Va	0.003 / 0.001	0.000 / 0.000	0.005 / 0.004
SE	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
Max Soft	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000

(b) Standardabweichungen

Tabelle 3.5: AUROCs und AUPRs untersuchter Uncertainty Estimates für die OOD Erkennung. Dargestellt sind die Mittelwerte und Standardabweichungen nach fünffacher Ausführung der Uncertainty Estimation Methoden. Die jeweils besten Mittelwerte sind hervorgehoben.

Dass die NUC-Methoden allerdings auch gut für die OOD Erkennung geeignet sein können, bestätigen die Ergebnisse für Cifar100. Hier haben beide Versionen die besten AUROCs und AUPRs. Jedoch ist deren Performance für den Cifar10 Datensatz schlechter als die der Ensemble Methoden und des MC Dropout Ansatzes, obwohl genug Daten und auch inkorrekte Prädiktionen für das Training der externen Netzwerke vorhanden sind.

Für den Cifar10 Datensatz erreicht die prädiktive Entropie des MC Dropouts (MCD PE) die besten Ergebnisse und für Cars die prädiktive Entropie des Data Augmentation Ensembles (DA PE). Fast jede Mutual Information und prädiktive Entropie der Ensemble Methoden und des MC Dropouts ist für die OOD Erkennung auf den Datensätzen Cifar10

und Cars besser geeignet als die einfache Softmaxentropie, was bestätigt, dass Ensembles und das MC Dropout besser die epistemische Unsicherheit erfassen können als die Softmaxausgabe. Nur für Cifar100 ist die Softmaxentropie besser. Wie bereits erwähnt, wird vermutet, dass die Softmaxausgabe aufgrund der schlechten Accuracy des Modells hier eine gute Qualität für die OOD Erkennung hat.

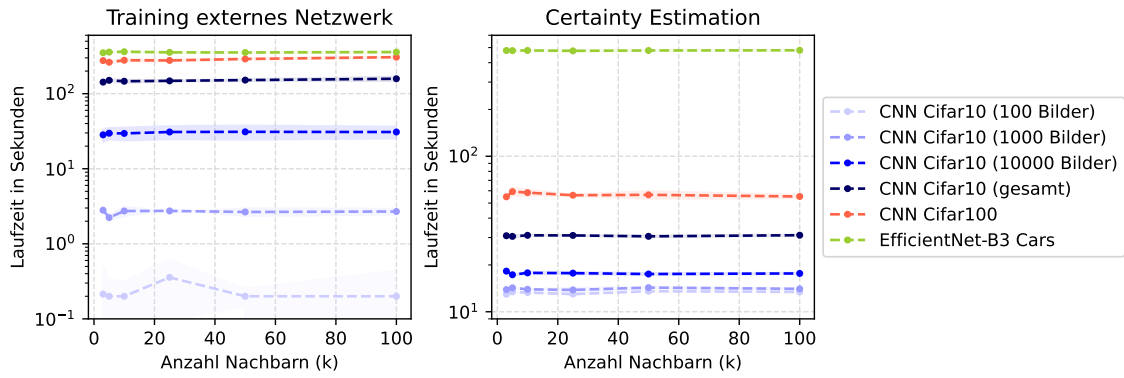
Abschließend ist festzuhalten, dass es für alle drei untersuchten Datensätze stets Uncertainty Estimation Methoden gibt, die besser für OOD Erkennung geeignet sind als die Verwendung der Softmaxausgabe, vor allem als die alleinige Nutzung des maximalen Softmaxscores.

3.3.4 Laufzeitvergleich der Methoden

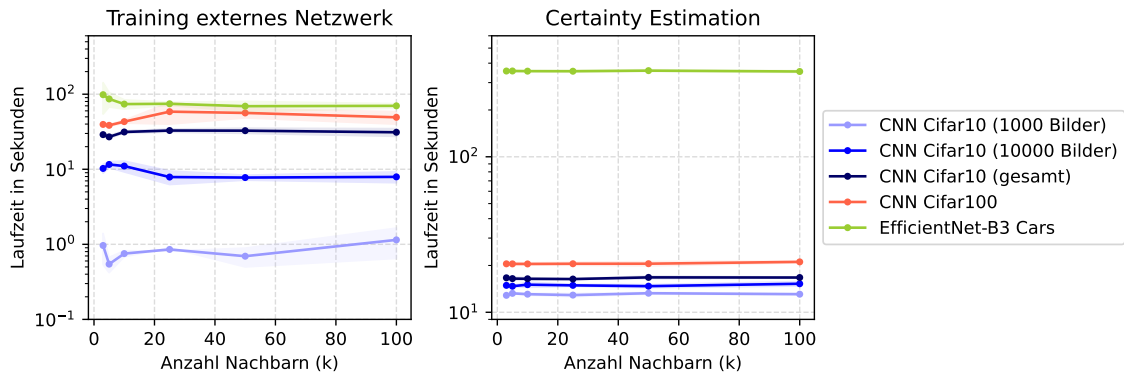
Abschließend werden die Laufzeiten der einzelnen Methoden betrachtet. Dabei wird zwischen der Vorbereitung und der Anwendung der Methoden unterschieden. Zur Vorbereitung zählt das Training der Ensemblemitglieder und des externen Netzwerks vom Neighborhood Uncertainty Classifier (NUC) sowie, falls notwendig, die Ermittlung der isotonischen Regressionsfunktion. Die ermittelte Regressionsfunktion kann anschließend für die Anwendung zur Testzeit gespeichert werden. Jede Uncertainty Estimation Methode wird dabei analog zu den vorigen Tests circa fünfmal neu vorbereitet und ausgeführt, sodass die ungefähr zu erwartende Laufzeit bestimmt werden kann. Die Zeitmessung für die Anwendung der Methoden wird vor der Ermittlung einer Prädiktion gestartet, weil diese oft Teil der Uncertainty Estimation ist. Das ist zum Beispiel beim NUC der Fall. Außerdem beziehen sich Uncertainty Estimates verschiedener Methoden teilweise auf unterschiedliche Prädiktionen. Die Ausgabe der Ensemble Methoden ist eine Prädiktion des gesamten Ensembles und bei der einfachen Softmaxentropie die Prädiktion eines einzelnen Modells.

Ob der Parameter k der NUC-Methoden Einfluss auf die Laufzeit hat, wird ebenfalls untersucht. Er gibt an, wie viele nächstgelegene Nachbarn im Repräsentationsraum eines DNNs ausgewertet werden, um einen Certainty Score zu berechnen. Da das CNN c10₁₀₀ zu wenig Validierungsdaten hat, um ein Modell sinnvoll auf ihnen zu trainieren, wird für das CNN c10₁₀₀ auf diese Version des NUC-Trainings (NUC Va) verzichtet. Die Zeiten wurden bei Ausführung auf einem Prozessor, der 64 Kerne enthält, gemessen.

Abbildung 3.12 zeigt die Laufzeiten für die NUC-Methoden in Abhängigkeit vom Parameter k . Für alle $k \in \{3, 5, 10, 25, 50, 100\}$ wurde die Laufzeit bestimmt. Die Laufzeiten scheinen für beide Versionen des NUCs in etwa von k unabhängig zu sein. Stattdessen erhöhen die Größe des beim Training verwendeten Datensatzes und die Komplexität des Modells die Laufzeit beim Training und bei der Certainty Estimation zur Testzeit. Dass die für das Training verwendeten Daten einen Einfluss auf die Laufzeit zur Testzeit haben, liegt daran, dass deren Repräsentationsvektoren im NUC gespeichert sind und zur Ermittlung der nächstgelegenen Nachbarn verwendet werden. Dies hat eine höhere Anzahl an Rechenoperationen bei der Ermittlung der Nachbarn zur Folge. Beim Vergleich der beiden NUC-Methoden sieht man, dass die Laufzeiten von NUC Va in Abbildung 3.12b stets kleiner sind als die von NUC Tr in Abbildung 3.12a, denn die genutzten Validierungsdatensätze sind kleiner als die Trainingsdatensätze.



(a) Laufzeiten für Certainty Netzwerke trainiert auf Trainingsdaten



(b) Laufzeiten für Certainty Netzwerke trainiert auf Validierungsdaten

Abbildung 3.12: Logarithmisch skalierte Darstellung der Laufzeiten für das Training des externen Netzwerks und für die anschließende Certainty Estimation als Funktion der Anzahl Nachbarn k . Betrachtet werden die externen Netzwerke für verschiedene Modelle. Visualisiert sind die Mittelwerte nach fünffachem Training des Certainty Netzwerks und die Standardabweichungen (leicht schattiert).

Tabelle 3.6 gibt einen Überblick der durchschnittlichen Laufzeiten aller untersuchten Uncertainty Estimation Methoden für jedes betrachtete neuronale Netzwerk auf einem Prozessor mit 64 Kernen. Mit Abstand am schnellsten ist die Abschätzung der prädiktiven Unsicherheit wenig überraschend mit der einfachen Softmaxentropie. Die Laufzeit entspricht in etwa der Laufzeit des Feedforward Laufs zur Ermittlung der Prädiktion. Somit erfolgt die Ausgabe einer Abschätzung für die prädiktive Unsicherheit ohne eine merklich höhere Laufzeit nebenbei zur Berechnung der Prädiktion.

Während die Laufzeit zur Berechnung von Certainty Scores der NUC-Methode mit wachsender Anzahl genutzter Trainingsdaten steigt, sind die Feedforward Durchläufe der Ensembles und des MC Dropouts zur Testzeit unabhängig von den Trainingsdaten und ihre Laufzeiten bleiben in etwa konstant. Auf die Laufzeit der Feedforward Durchläufe zur Testzeit hat jedoch die Komplexität des Netzwerks einen starken Einfluss, weshalb die NUC-Methoden für das EfficientNet deutlich schneller sind. Sie führen nur einen Feedforward Lauf pro Prädiktion aus, die Ensembles, die aus 5 Mitgliedern bestehen, hingegen 5. Die Uncertainty Estimation durch ein Ensemble ist nochmals deutlich schneller als das MC Dropout, da hier für diese Methode 50 Feedforward Läufe ausgeführt werden, wie von Y. Gal [Gal16] empfohlen wird.

	CNN c10 ₁₀₀	CNN c10 ₁₀₀₀	CNN c10 ₁₀₀₀₀	CNN c10	CNN c100	Eff Cars
Softmaxentropie (SE)						
Kal.	0.1 s	0.3 s	0.6 s	2.3 s	2.3 s	60.6 s
<i>Est.</i>	<i>1.8 s</i>	<i>1.9 s</i>	<i>1.9 s</i>	<i>1.9 s</i>	<i>2.1 s</i>	<i>3.5 min</i>
Monte Carlo Dropout (PE und/oder MI)						
Kal.	0.9 s	4.2 s	12.4 s	20.4 s	20.5 s	16 min
<i>Est.</i>	<i>19.8 s</i>	<i>19.8 s</i>	<i>19.9 s</i>	<i>19.5 s</i>	<i>19.9 s</i>	<i>110 min</i>
Bagging Ensemble (PE und/oder MI)						
Tr.	4.7 s	18.5 s	3.7 min	20 min	23 min	71 h
Kal.	0.6 s	2.8 s	10.1 s	13.7 s	15.0 s	2.4 min
<i>Est.</i>	<i>13.8 s</i>	<i>14.1 s</i>	<i>13.9 s</i>	<i>13.6 s</i>	<i>14.8 s</i>	<i>17 min</i>
Data Augmentation Ensemble (PE und/oder MI)						
Tr.	13.8 s	79.8 s	16 min	60 min	63 min	83 h
Kal.	0.7 s	4.6 s	10.4 s	13.8 s	14.4 s	2.2 min
<i>Est.</i>	<i>13.5 s</i>	<i>13.3 s</i>	<i>13.5 s</i>	<i>13.5 s</i>	<i>13.4 s</i>	<i>17 min</i>
NUC trainiert auf Trainingsdaten (NUC Tr)						
Tr.	0.2 s	2.8 s	28.3 s	2.4 min	4.6 min	5.9 min
<i>Est.</i>	<i>13.0 s</i>	<i>13.9 s</i>	<i>18.2 s</i>	<i>30.8 s</i>	<i>54.9 s</i>	<i>8.0 min</i>
NUC trainiert auf Validierungsdaten (NUC Va)						
Tr.	-	1.0 s	10.3 s	28.9 s	39.6 s	98.5 s
<i>Est.</i>	-	<i>12.9 s</i>	<i>14.9 s</i>	<i>16.7 s</i>	<i>20.5 s</i>	<i>4.3 min</i>

Tabelle 3.6: Laufzeiten der Methoden zur Uncertainty Estimation bei Ausführung mit einem Prozessor, der 64 Kerne enthält. Für jedes betrachtete Modell wird die Zeit für das Training notwendiger Netzwerke (Tr.) zur Vorbereitung einzelner Methoden gemessen. Die Zeit zur Ermittlung der isotonischen Regressionsfunktion für die Kalibrierung der Uncertainty Estimates wird, falls notwendig, erfasst (Kal.). Die Zeiten für die eigentliche Uncertainty Estimation der Methoden ist optisch hervorgehoben (*Est.*). Dargestellt sind die Mittelwerte nach fünffacher Wiederholung.

Bei Betrachtung der Vorbereitungszeit der Methoden fällt auf, dass das Training zusätzlicher Netzwerke für die NUC-Methoden deutlich schneller erfolgt als für die Ensemble Methoden. Der NUC trainiert nur ein Netzwerk, was lediglich eine geringe Komplexität hat [RM19]. Unter den zwei Ensembles ist das Training der Mitglieder beim Bagging Ensemble stets schneller als das mit augmentierten Bildern, da letzteres die Trainingsbilder vor jeder Trainingsepoche erneut modifiziert. Vor allem für die komplexe EfficientNet-B3-Architektur dauert das Ensembletraining sehr lange, sogar mehrere Tage. Folglich sind Ensemble Methoden gegenüber dem Monte Carlo Dropout nur im Vorteil, wenn das Ensemble selten neu trainiert werden muss ³.

³Bemerke, die Laufzeit des Trainings kann beschleunigt werden, indem die einzelnen Mitglieder parallel trainiert werden. Dies ist jedoch mit einer Erhöhung der Ressourcen verbunden.

Die Laufzeit für das Berechnen der isotonischen Regressionsfunktion zur Kalibrierung der Estimates ist von der Menge verwendeter Daten für die Berechnung abhängig. Zur Ermittlung der Regressionsfunktion werden die Validierungsdaten verwendet. Die Größe des Validierungsdatensatzes steigt bei den Experimenten mit der Größe des Trainingsdatensatzes, weshalb ein Wachstum der Laufzeit für kalibrierte Estimates vom CNN c10₁₀₀ bis zum CNN c10 in Tabelle 3.6 erkennbar ist. Verglichen mit der Laufzeit des Trainings der Ensembles ist die zusätzliche Vorbereitungszeit durch die Bestimmung der isotonischen Regressionsfunktion sehr klein.

Zusammenfassend ist zu erwähnen, dass sich die Methoden stark in ihrer Laufzeit für die Anwendung zur Testzeit sowie in der benötigten Zeit zur Vorbereitung der Uncertainty Estimation unterscheiden. Faktoren, die die Laufzeiten erhöhen sind die Komplexität des Modells, die Anzahl ausgeführter Feedforward Läufe im Lauf der Methode und die Menge der Trainingsdaten.

4 Verwendung von Uncertainty Estimates für das aktive Lernen

Das aktive Lernen ist ein bedeutsamer Anwendungsbereich der Uncertainty Estimation. Ziel ist die Maximierung der Modellperformance, wobei so wenig Daten wie möglich beschriftet werden [RXC⁺21]. Das aktive Lernen verspricht somit den Aufwand für das Labeling drastisch zu reduzieren, was den Einsatz des aktiven Lernens bei Volkswagen motiviert. In diesem Kapitel wird zunächst der Ablauf des aktiven Lernens geklärt und erläutert, wie Uncertainty Estimates in diesem Prozess eingesetzt werden können. Es folgen Simulationen des aktiven Lernens mit Uncertainty Estimates der in Abschnitt 2.3 vorgestellten Methoden, wobei festgestellt werden soll, welche am geeignetsten sind und wie sie am besten eingesetzt werden können.

4.1 Skizzierter Ablauf des aktiven Lernens mit Uncertainty Estimates

Der grobe Ablauf des aktiven Lernens besteht zunächst aus der gezielten Auswahl der für das Lernen eines Modells nützlichsten Daten. Anschließend erfolgt das Labeling dieser durch ein Orakel, zum Beispiel durch Annotation von einem Menschen. Danach wird das Modell mit allen gelabelten Daten trainiert. Dieser Prozess kann mit dem neu dazugewonnen Wissen des Modells beliebig oft wiederholt werden. Abbruchbedingungen sind häufig vordefinierte Erwartungen an die Performance, zum Beispiel das Erreichen einer bestimmten Accuracy auf Testdaten. Für die Auswahl eines Samples gibt es verschiedene Herangehensweisen. Sie kann unabhängig vom Rest der vorliegenden Datenmenge geschehen, indem anhand bestimmter Eigenschaften des Samples entschieden wird, ob es einen hohen Mehrwert für das Training des Modells hat. Eine weitere Möglichkeit ist die Auswahl eines Samples basierend auf einem Ranking der gesamten ungelabelten Datenmenge. Letzteres, das sogenannte Pool-basierte aktive Lernen, wird häufiger eingesetzt, beispielsweise beim Deep Active Learning (DeepAL). Die Auswahl eines einzelnen Samples, wie es beim traditionellen aktiven Lernen der Fall ist, ist für das Training von Deep Neural Networks (DNNs) sehr ineffizient. Daher verwendet man für DNNs DeepAL, welches auf batch-basierten Auswahlstrategien basiert, wobei mehrere Daten gleichzeitig ausgewählt und vom Orakel gelabelt werden. [RXC⁺21]

Die Auswahlstrategie des DeepAL kann folgendermaßen formal definiert werden. Sei U die Menge ungelabelter Samples. U ist ein Datenpool, zu dem stets neue Daten hinzugefügt werden können [Sie20]. $L = \{\mathbf{X}, \mathbf{Y}\}$ sei der aktuelle Trainingsdatensatz, bestehend aus den Samples \mathbf{X} und den zugehörigen Labels \mathbf{Y} . Die Domäne der Samples sei \mathcal{X} und die der Labels \mathcal{Y} . Mittels der Bewertungsfunktion $Q_f : \mathcal{X}^j \rightarrow \mathbb{R}$ wird unter Verwendung des

DNNs $f : \mathcal{X} \rightarrow \mathcal{Y}$ entschieden, welche Samples für das Training von f besonders geeignet sind:

$$\{\mathbf{x}_1, \dots, \mathbf{x}_b\} = \operatorname{argmax}_{\{\mathbf{x}_1^*, \dots, \mathbf{x}_b^*\} \subseteq U} Q_f(\mathbf{x}_1^*, \dots, \mathbf{x}_b^*). \quad (4.1)$$

$S = \{\mathbf{x}_1, \dots, \mathbf{x}_b\}$ ist der ausgewählte Samplebatch der Größe b .

Das DeepAL eignet sich für das Training der DNNs bei Volkswagen. Die Datenbasis bilden verschiedene Bilddaten. Die Eigenschaft anhand welcher die Bilder bewertet werden, ist die prädiktive Unsicherheit des Modells, die mittels Uncertainty Estimation approximiert wird. Bei dieser Auswahlstrategie werden die Samples mit den höchsten Uncertainty Estimates aus U für das Labeling ausgewählt:

$$Q_f(\mathbf{x}_1^*, \dots, \mathbf{x}_b) = \sum_{i=1}^b \operatorname{uncert}(f(\mathbf{x}_i^*)). \quad (4.2)$$

Motivation hierfür ist, dass eine hohe prädiktive Unsicherheit mit einer geringeren Wahrscheinlichkeit für die Korrektheit der Prädiktion korrespondiert. Wiederum kann ein Netzwerk von Daten, für die es inkorrekte Prädiktionen berechnet, vermutlich noch am meisten lernen. Beim anschließenden Training mit diesen Daten werden falsche Prädiktionen mit einem hohen Fehler bestraft und eine Verbesserung der Modellperformance für diese und ähnliche Samples ist zu erwarten. Ist die vorhergesagte prädiktive Unsicherheit für ein Sample hingegen besonders gering, so ist die Prädiktion wahrscheinlich korrekt. Das Labeling des Samples wäre redundant, da das Netzwerk schon über genug Wissen ähnlicher Samples verfügt. Das Einbeziehen eines solchen Samples in den Trainingsprozess würde weniger dazu beitragen, dass das Netzwerk neue Testdaten korrekt klassifiziert als ein Sample, dessen prädiktive Unsicherheit hoch ist.

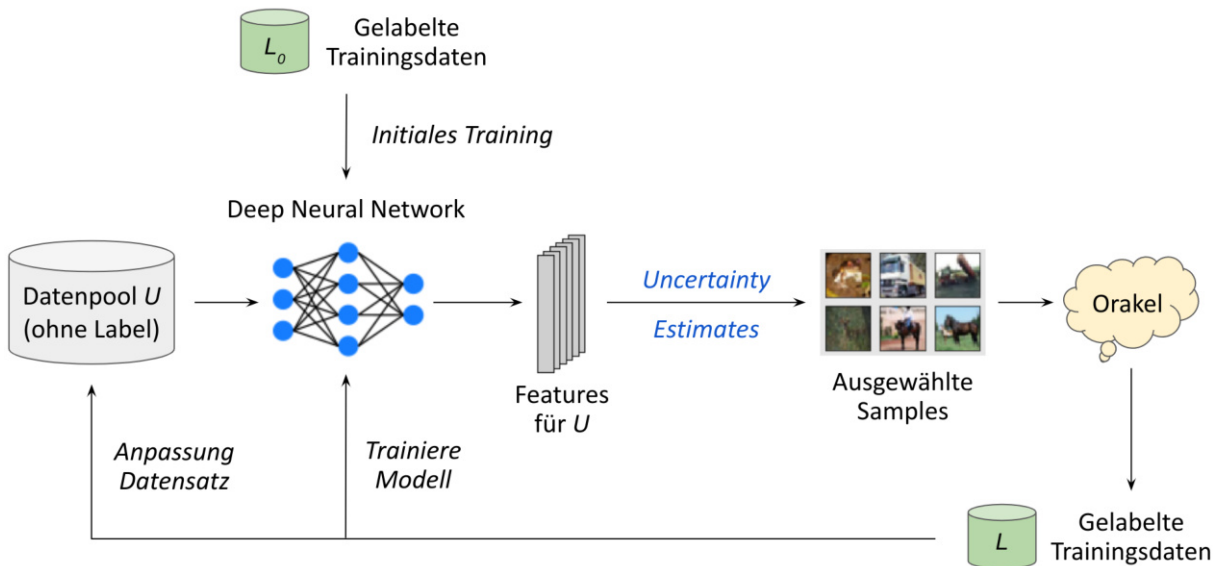


Abbildung 4.1: Kreislauf des aktiven Lernens mit Uncertainty Estimates als Rankingkriterium nach [RXC⁺21].

Abbildung 4.1 skizziert den Kreislauf des aktiven Lernens mit Uncertainty Estimates. Das DNN wird zunächst auf bereits vorhandenen, gelabelten Trainingsdaten L_0 vortrainiert und die Samples des ungelabelten Datenpools U werden verwendet, um aus den Feedforward Durchläufen des DNNs Informationen (Features) für die Abschätzung der

prädiktiven Unsicherheit zu extrahieren. Anhand ihres Uncertainty Estimates werden die vom Netzwerk am wenigsten verstandenen Samples für das anschließende Labeling durch das Orakel ausgewählt. Die gelabelten Samples werden dem Trainingsdatensatz L hinzugefügt und gleichzeitig aus dem Pool U entfernt. Falls vorhanden werden neue Daten zu U hinzugefügt. [RXC⁺21]

Unsicherheitsbasierte Auswahlstrategien werden häufig für das aktive Lernen verwendet [RXC⁺21]. Jedoch sind dies wegen ihres geringen Rechenaufwands hauptsächlich einfache Techniken, die sich lediglich auf eine einzelne Ausgabe des Netzwerks beziehen. Zu solchen Methoden zählt für Klassifizierungsnetzwerke die Softmaxentropie [RXC⁺21], die in Abschnitt 2.4 definiert wird. Möglicherweise können jedoch andere Metriken für die gezielte Auswahl weiterer Trainingsdaten besser geeignet sein. Die Softmaxentropie erfasst hauptsächlich die aleatorische Unsicherheit in den Daten und nur teilweise die epistemische. Die aleatorische Unsicherheit kann auch nach weiterem Training nicht verkleinert werden. Ein Beispiel für ein Sample mit hoher aleatorischer Unsicherheit ist ein Bild, auf dem in der Ferne ein Motorrad abgebildet ist, das jedoch so klein ist, dass es sogar für den Menschen leicht mit einem Fahrrad zu verwechseln ist. Unsicherheit besteht hier zwischen den beiden Klassen Fahrrad und Motorrad. Fügt man das Sample mit seinem Label Motorrad dem Trainingsdatensatz hinzu, so lernt das Netzwerk die Klasse Motorrad vorherzusagen. Bei ähnlichen Bildern könnte in der Ferne jedoch ein Fahrrad abgebildet sein, was nun falsch klassifiziert wird. Das Erlernen von Informationen aus Daten hoher aleatorischer Unsicherheit kann somit die Performanceverbesserung des Modells behindern. Aus diesem Grund wird in diesem Kapitel jede der untersuchten Uncertainty Estimation Methoden für das aktive Lernen getestet. Eine graphische Darstellung der aleatorischen und epistemischen Unsicherheit befindet sich in Abschnitt 2.2.3.

Bei der Bewertung der Uncertainty Estimates ungelabelter Daten ist zu beachten, dass das Ziel einiger Methoden nicht die Abschätzung der prädiktiven Unsicherheit für ein einzelnes Netzwerk ist. Ensemble Methoden verwenden zum Beispiel mehrere verschiedene Netzwerke. Uncertainty Estimates dieser Methode beziehen sich auf die Prädiktion des Ensembles, also auf das arithmetische Mittel der Ausgaben aller Ensemblemitglieder. Um Ensemble Methoden dennoch im Bereich des aktiven Lernens einzusetzen, wird für jedes Ensemble Mitglied exakt die Modellarchitektur des DNNs f verwendet, das mit aktivem Lernen trainiert werden soll. Die aktuellen Trainingsdaten L bilden die Datenbasis für das Training der Ensemblemitglieder. Auf L können, wie in Kapitel 2.3.4 beschreiben, Techniken wie Bagging und Data Augmentation für die Ensembles angewandt werden. Prädiktionen, bei denen sich ein solches Ensemble unsicher ist, können aufgrund der Parallelen der Ensemblemitglieder zu f auch für das Training von f hilfreich sein.

Beim Monte Carlo (MC) Dropout beziehen sich die Uncertainty Estimates auf den MC Dropout Schätzer, den Erwartungswert für die prädiktive Wahrscheinlichkeitsverteilung von f bei aktivem Dropout. Der Schätzer ist das arithmetische Mittel mehrerer Feedforward Läufe des Netzwerks mit aktivierten Dropout Schichten (vergleiche Kapitel 2.3.3). Die Uncertainty Estimates beziehen sich somit zwar nicht direkt auf die typischerweise mit deaktivierten Dropout berechnete Modellausgabe für ein Testbeispiel, sollten jedoch für das anschließende Training gut geeignet sein, da Dropout beim Training stets aktiviert ist. Y. Gal [Gal16] konnte bereits für den MNIST Datensatz zeigen, dass sich die Modellperformance durch das aktive Lernen unter Nutzung des MC Dropouts deutlich schneller erhöht als es bei der zufälligen Wahl neuer Trainingsdaten der Fall ist.

4.2 Bewertung des unsicherheitsbasierten DeepAL am Beispiel von Cifar10

Welche Uncertainty Estimation Methoden sich am besten für das DeepAL eignen, wird nun für die Klassifizierung mit dem Cifar10 Datensatz getestet. Der in Abbildung 4.1 skizzierte Kreislauf wird hierbei für ein Convolutional Neural Network simuliert. Da Cifar10 bereits vollständig gelabelt ist, werden die Bilder aus den Cifar10 Trainingsdaten von ihrem Label getrennt und als ungelabelter Datenpool U verwendet. Werden Daten aus diesem Pool für das Training des Modells ausgewählt, wird diesen Bildern ihr bekanntes Label zugewiesen. Das ersetzt den Schritt des Labelings durch ein Orakel.

Wie bereits in Kapitel 3.1 erläutert, wird für das Training von DNNs auf eigenen Daten fast immer Transfer-Lernen verwendet. So auch bei Volkswagen. Die Modelle haben meistens zu viele Parameter, um sie mit wenigen vorhandenen Daten zu trainieren, weshalb die Modellparameter ähnlicher Domänen verwendet werden. Da für das DeepAL besonders wenige gelabelte Daten vorliegen, kann das Transfer-Lernen hier einen sehr großen, positiven Einfluss auf die Modellperformance haben. Daher wird für die Simulation in diesem Abschnitt ebenfalls Transfer-Lernen verwendet. Der Cifar100 Datensatz ist dem Cifar10 Datensatz sehr ähnlich, sodass der gesamte Cifar100 Datensatz für das Transfer-Lernen genutzt wird. Das DNN f hat die Architektur des CNN c10, das bereits in Kapitel 3.1 beschrieben wurde. Details zum Transfer-Lernen können ebenfalls diesem Kapitel entnommen werden.

Unterschieden wird zwischen verschiedenen Größen des initialen Trainingsdatensatzes L_0 und unterschiedlich vielen Bildern die pro Iteration dem Trainingsdatensatz hinzugefügt werden. Das Netzwerk ist zum Startpunkt entweder kaum vortrainiert, $|L_{01}| = 100$, bereits ein wenig vortrainiert, $|L_{02}| = 1000$, oder hat schon einen recht großen Trainingsdatensatz, $|L_{03}| = 10000$. Es handelt sich hierbei um die in Kapitel 3 betrachteten Netzwerke CNN c10₁₀₀, CNN c10₁₀₀₀ und CNN c10₁₀₀₀₀. Für L_{01} werden pro Iteration die 100 Bilder mit den höchsten Uncertainty Estimates und ihren Labels ($b = 100$) dem Trainingsdatensatz L hinzugefügt. Ist L_{02} der initiale Trainingsdatensatz, so wird neben der kleinen Batch-Größe von 100 Daten pro Iteration auch das Hinzufügen von 1000 Daten pro Iteration betrachtet. Da im Fall von L_{03} nach einer Vergrößerung des Trainingsdatensatzes um je 100 Bilder kaum ein Unterschied in der Performance des Modells zu erwarten ist, werden hier nur große Batches mit 1000 Bildern, die pro Iteration hinzugefügt werden, untersucht. Je größer der ausgewählte Batch, desto weniger Iterationen erfolgen in der Praxis bei einem konstanten Labelaufwand, weshalb das Netzwerk seltener neu trainiert werden muss.

Die Validierungsdatensätze der Simulationen haben jeweils dieselbe Größe wie die initialen Trainingsdatensätze. Sie enthalten somit mehr Bilder als die eigentlich bei Volkswagen verwendete Anzahl von 20% der Größe des Trainingsdatensatzes. Jedoch bleibt der Validierungsdatensatz für die Simulation unverändert, während der Trainingsdatensatz L wächst. Da die von der Auswahlstrategie Q_f ausgewählten Bilder für das Modelltraining bestimmt sind, wird auf das Hinzufügen von Teilen dieser Bilder zum Validierungsdatensatz verzichtet. In der Praxis können zufällige Samples aus dem Pool U gelabelt und zum Validierungsdatensatz hinzugefügt werden, falls dieser vergrößert werden soll. Beim Training des Modells wird wie in Kapitel 3 Early Stopping und die Reduzierung der Lernrate verwendet.

Zu Beginn jedes Trainings wird die Lernrate wieder leicht erhöht, sodass die Modellparameter ein neues lokales Optimum auf dem durch den erweiterten Trainingsdatensatz veränderten Graphen der Lossfunktion erreichen können.

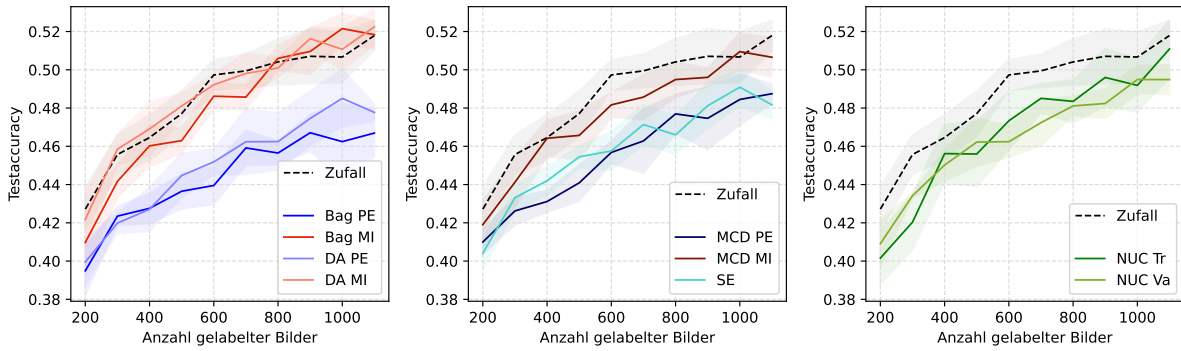
Damit die Performanceveränderung des Netzwerks über den Prozess des aktiven Lernens hinweg bewertbar ist, wird nach jeder Iteration die Accuracy des Modells für den unabhängigen Testdatensatz erfasst. Insgesamt erfolgt die Durchführung von 10 Iterationen. Je schneller die Testaccuracy im Verlauf der Iterationen wächst, desto besser ist die verwendete Uncertainty Estimation Methode für das DeepAL geeignet. Das Modelltraining ist sehr stochastisch, weshalb der Prozess des aktiven Lernens für jede Uncertainty Estimation Methode fünfmal durchgeführt wird und Mittelwerte sowie Standardabweichungen der Testaccuracy ausgewertet werden.

Die Uncertainty Estimates folgender Methoden und Metriken werden betrachtet: die Monte Carlo Dropout Methode mit den Metriken prädiktive Entropie (MCD PE) und der Mutual Information (MCD MI), das Bagging Ensemble und das Data Augmentation Ensemble ebenfalls mit den Metriken prädiktive Entropie (Bag PE, DA PE) und Mutual Information (Bag MI, DA MI), die Softmaxentropie der Netzwerkausgabe (SE) sowie der Neighborhood Uncertainty Classifier, wobei das externe Netzwerk auf dem Trainingsdatensatz L (NUC Tr) oder dem Validierungsdatensatz (NUC Va) trainiert wird und $k = 3$ Nearest Neighbors verwendet werden. Zusätzlich wird der Fall betrachtet, dass neue Daten zufällig für das Labeling ausgewählt und zu L hinzugefügt werden. Somit kann der Einfluss der gezielten Auswahl neuer Trainingsdaten durch das unsicherheitsbasierte DeepAL abgeschätzt werden.

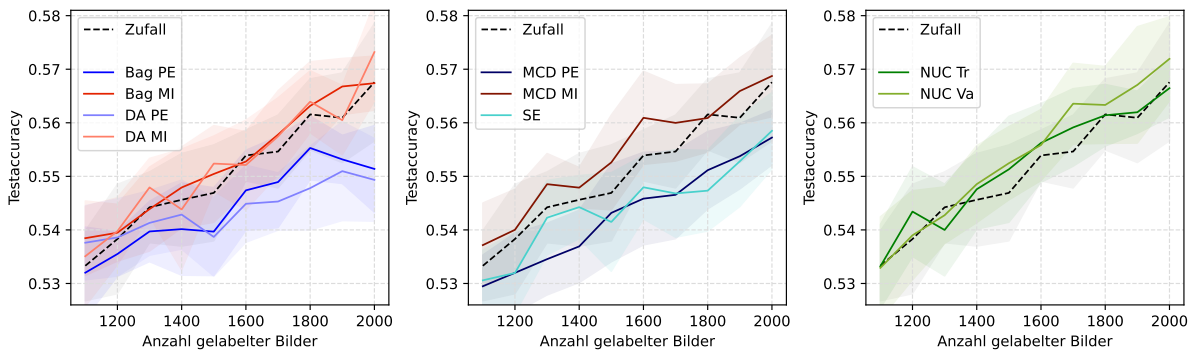
Die Ergebnisse für die Batch-Größe $b = 100$ sind in Abbildung 4.2 dargestellt. Zur besseren Übersichtlichkeit sind die Graphen auf mehrere Diagramme verteilt. Kurven ähnlicher Methoden sind zusammen in einem Diagramm visualisiert. Überraschender Weise steigt die Testaccuracy, wenn f lediglich auf 100 Bildern vortrainiert wurde (Abbildung 4.2a), mit am schnellsten, wenn neue Daten zufällig ausgewählt werden. Die besten Ergebnisse von den Uncertainty Estimation Methoden erreicht die Mutual Information der Ensemble Methoden, Bag MI und DA MI (linkes Diagramm in Abbildung 4.2a). Jedoch verlaufen die Kurven nicht deutlich anders als die Kurve zufälliger Samples und das Training eines Ensembles ist mit einer hohen Laufzeit verbunden.

Wie bereits vermutet sind die Resultate der Entropien am schlechtesten, da die Trainingsdaten um Daten mit hoher aleatorischer Unsicherheit erweitert werden. Für die Softmaxentropie (SE), die häufig für das unsicherheitsbasierte DeepAL verwendet wird, konnten D. Wang et al. [WS14] ebenfalls feststellen, dass sie oft zu schlechteren Ergebnissen als die zufällige Auswahl führt. Warum jedoch fast alle Uncertainty Estimates für das aktive Lernen des kaum vortrainierte CNNs derart schlechtere Ergebnisse als zufällige Samples aufweisen, wird am Ende dieses Abschnitts geklärt.

Bezüglich Abbildung 4.2b ist teilweise eine leichte Verbesserung der Testaccuracy durch das Hinzufügen von Daten mit hoher prädiktiver Unsicherheit im Vergleich zu zufälligen Samples bemerkbar. Die Mutual Information des MC Dropouts (MCD MI) und der Ensemble Methoden sowie die NUC-Methoden erreichen geringfügig bessere Ergebnisse als zufällige Samples. Zum Beispiel wird die Testaccuracy von 56% am schnellsten mit Uncertainty Estimates der MCD MI erreicht. Diese Accuracy ist hier beim Training mit circa 1600 Bildern zu erwarten und bei zufälligen Samples werden durchschnittlich 1800 Trainingsdaten benötigt. Dies entspricht einem um 11.1% geringeren Labelaufwand durch



(a) Entwicklung der Testaccuracy eines auf 100 Bildern vortrainierten CNNs



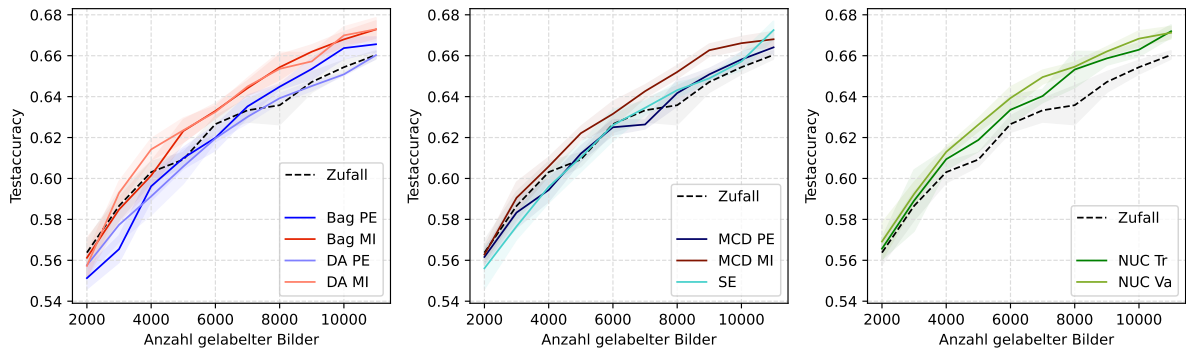
(b) Entwicklung der Testaccuracy nach dem Start mit 1000 Bildern

Abbildung 4.2: Testaccuracy als Funktion der Größe des Trainingsdatensatzes für die initialen Trainingsdatensätze L_{01} und L_{02} . Es wurden 10 Mal 100 Daten hinzugefügt. Die Auswahl erfolgte anhand von Uncertainty Estimates verschiedener Methoden. Die unsicherheitsbasierte Auswahlstrategie steht hier im direkten Vergleich mit der zufälligen Auswahl neuer Daten. Standardabweichungen sind schattiert gekennzeichnet.

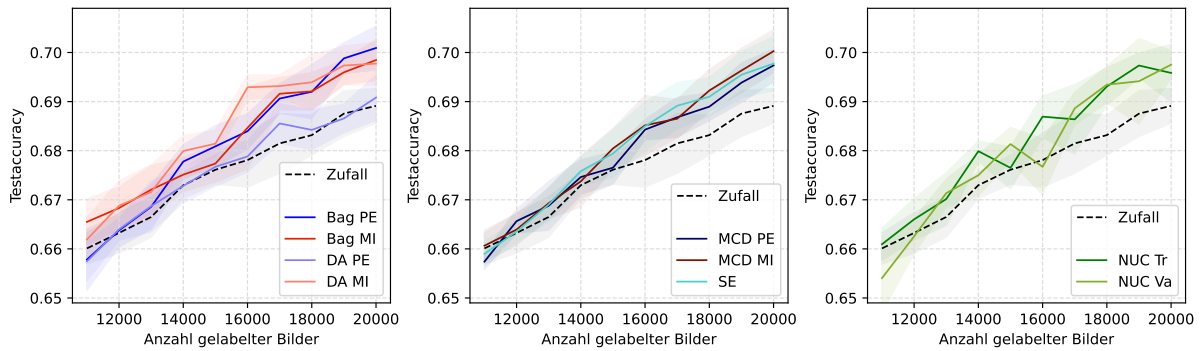
MCD MI. Jedoch ist dieser Unterschied unter Beachtung der hohen Standardabweichungen kaum relevant. Für die Entropien ist erneut zu erkennen, dass sie schlechtere Resultate als zufällige Samples aufweisen.

Abbildung 4.3 zeigt die Ergebnisse für das Hinzufügen von 1000 Daten pro Iteration, wobei in Abbildung 4.3a mit einem initialen Trainingsdatensatz von 1000 Bildern (L_{02}) und in Abbildung 4.3b mit 10000 Bildern (L_{03}) begonnen wird. In diesen Diagrammen ist deutlich zu erkennen, dass das unsicherheitsbasierte DeepAL zu einer Performanceerhöhung im Vergleich zu beliebig gewählten Trainingsdaten führen kann. In Abbildung 4.3a verlaufen die Funktionen der NUC Methoden und jeder Mutual Information großteils über den Graphen der zufälligen Samples. Positiv fällt auch die geringe Standardabweichung des Verlaufs der Graphen auf. Betrachtet man, ab wann eine Testaccuracy von 66% erreicht wird, so ist dies bei diesen Uncertainty Estimation Methoden nach circa 9000 Bildern der Fall und bei den zufälligen Samples erst bei einem Trainingsdatensatz der Größe 11000. Somit müssen durch die unsicherheitsbasierte Auswahl circa 18.2% weniger Daten gelabelt werden.

Die Ergebnisse des bereits gut vortrainierten Netzwerks in Abbildung 4.3b zeigen ein deutlich schnelleres Wachstum der Testaccuracy durch die Auswahl von Samples mit



(a) Entwicklung der Testaccuracy eines auf 1000 Bildern vortrainierten CNNs



(b) Entwicklung der Testaccuracy nach dem Start mit 10000 Bildern

Abbildung 4.3: Darstellung der Testaccuracy als Funktion der Größe des Trainingsdatensatzes für die initialen Trainingsdatensätze L_{02} und L_{03} . Am Ende jeder Iteration wurden 1000 Bilder zu L hinzugefügt. Die Auswahl erfolgte anhand von Uncertainty Estimates verschiedener Methoden oder zufällig. Standardabweichungen sind schattiert gekennzeichnet.

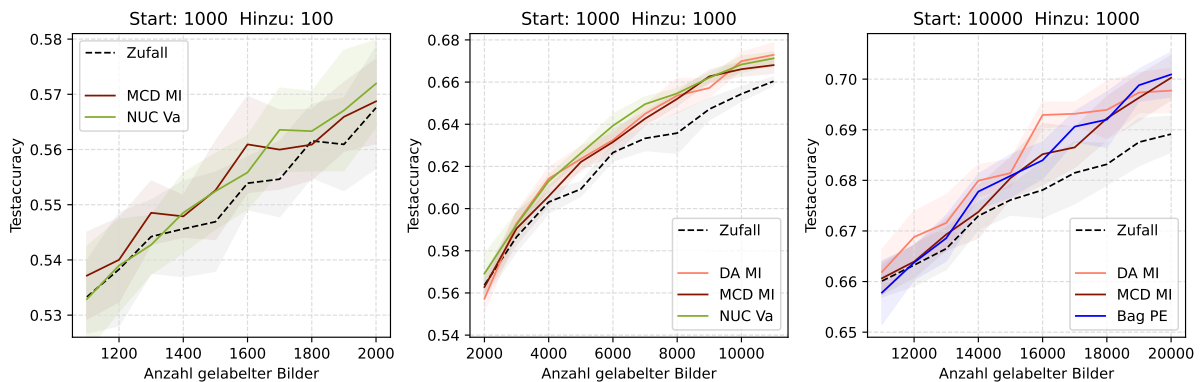


Abbildung 4.4: Darstellung der besten Graphen aus den Abbildungen 4.2b, 4.3a und 4.3b sowie der Graphen für zufällige Samples.

hohen Uncertainty Estimates und das sogar für die meisten Entropien. Die Testaccuracy von 69% wird bei der zufälligen Auswahl im Durchschnitt nach 20000 Trainingsdaten erreicht. Durch die Mutual Information des Data Augmentation Ensembles (DA MI) werden lediglich 16000 Daten benötigt, was 20% weniger Bilder sind.

Die besten Graphen aus allen drei Diagrammen der Abbildungen 4.2b, 4.3a und 4.3b sind zusammen in Abbildung 4.4 dargestellt, um leichter festzustellen, welche Uncertainty Estimation Methode die besten Ergebnisse erreicht. Ist der Trainingsdatensatz L noch recht klein, so sind MCD MI und NUC Va am besten geeignet. Allerdings ist dies aufgrund der hohen Varianzen nicht eindeutig klar. Ist das Netzwerk f bereits gut vortrainiert, so ist die Methode DA MI gut geeignet. Es kann daher nicht genau bestimmt werden, welche Methode am besten für das unsicherheitsbasierte DeepAL geeignet ist. Jedoch sollten die NUC-Methoden und die Metrik Mutual Information der prädiktiven Entropie vorgezogen werden. Letztere erreicht erst nach sehr langem Training ein gutes Performancewachstum, wie im rechten Diagramm am Beispiel von Bag PE zu sehen ist. Folglich ist die Qualität der Uncertainty Estimates anhand der Metriken AUROC und AUPR kein gutes Maß für deren Performance beim aktiven Lernen, da die prädiktiven Entropien in Abschnitt 3.3.2 meistens bessere AUROCs und AUPRs aufweisen als die entsprechenden Uncertainty Estimates der Mutual Information.

Die unsicherheitsbasierte Auswahlstrategie hat für das aktive Lernen einen erheblichen Nachteil, der erklärt, weshalb die Modellperformance durch die Wahl zufälliger Samples anfangs schneller steigt als es durch unsichere Samples der Fall ist. Das Auswahlverfahren betrachtet keine Zusammenhänge zwischen den Samples. Jedes einzelne Bild mit einer hohen prädiktiven Unsicherheit ist zwar sehr nützlich für das anschließende Training, jedoch tendieren die in einer Iteration ausgewählten Samples dazu, sehr ähnlich zu sein. Die Informationen, die sie dem Modell bereitstellen sind fast identisch, was sowohl Label-Ressourcen verschwendet als auch die gute Generalisierung des Modells behindert [RXC⁺21]. Ein extremes Beispiel ist, dass dem Modell die Erkennung einer Klasse besonders schwerfällt, daher Bilder dieser Klasse eine sehr hohe prädiktive Unsicherheit haben und für das Labeling ausgewählt werden. Anschließend befinden sich deutlich mehr Bilder dieser Klasse im Trainingsdatensatz als vorher, während die Anzahl Samples anderer Klassen konstant bleibt. Das führt besonders bei kleinen initialen Datensätzen anfangs zu einer verfälschten Auffassung des Modells über die Wahrscheinlichkeitsverteilung der Daten. Anzumerken ist, dass dieser Nachteil, je kleiner die Größe des Datenpools U ist, umso weniger schwerwiegend ist. Die Samples im ausgewählten Batch sind so weniger ähnlich, da es in U insgesamt weniger Samples gibt.

Beim Start mit 100 Trainingsdaten und 100 Validierungsdaten sind 49800 ungelabelte Daten im Pool U . Abbildung 4.5 zeigt die Ergebnisse der erneuten Simulation des Prozesses des DeepAL mit L_{01} und der Batch-Größe 100, wenn U lediglich 10000 Daten enthält. Zwar ist vor allem bei der Mutual Information von MC Dropout und bei den NUC-Methoden eine Verbesserung gegenüber Abbildung 4.2a zu vermerken, jedoch ist noch immer keine Uncertainty Estimation Methode merklich besser geeignet als die zufällige Auswahl von Samples.

Bei den Experimenten von Y. Gal [Gal16] hingegen erwies sich die Verwendung von Uncertainty Estimates des MC Dropouts für das aktive Lernen stets als besser als die zufällige Auswahl der Samples. Dies kann mit der sehr kleinen Batch-Größe zusammenhängen. Pro Iteration wurden von Y. Gal lediglich 10 Bilder ausgewählt, weshalb dem Trainingsdatensatz stets nur wenig ähnliche Bilder hinzugefügt werden. Die Wahl solcher kleiner Batches ist für große Netzwerke ineffizient. Der Aufwand für das Labeling von 10 Bildern für ein Klassifizierungsproblem ist sehr gering, vor allem im Vergleich zur Laufzeit des anschließenden Trainings. Außerdem ist die Klassifizierung auf dem MNIST-Datensatz deutlich leichter. DNNs können hier bereits mit sehr wenigen Trainingsdaten

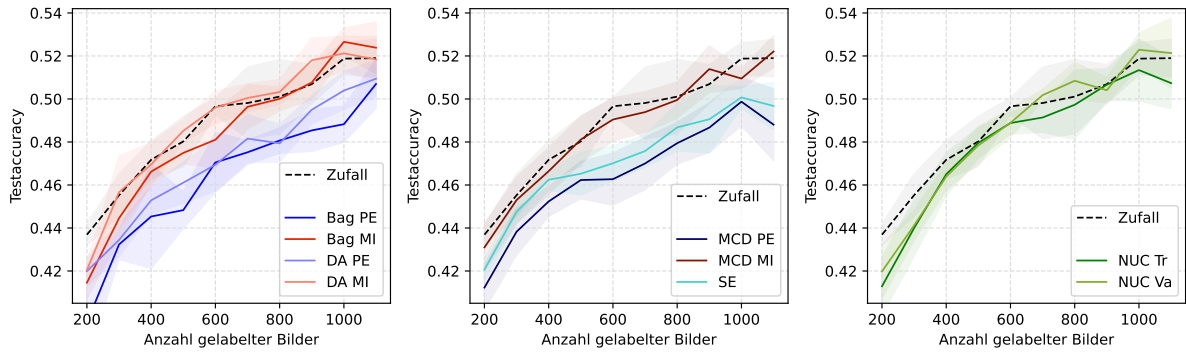


Abbildung 4.5: Testaccuracy als Funktion der Größe des Trainingsdatensatzes für den initialen Trainingsdatensatz L_{01} und einer verkleinerten Menge ungelabelter Daten mit $|U| = 10000$. Es wurden 10 Mal 100 Daten hinzugefügt. Die Auswahl der Daten erfolgte anhand von Uncertainty Estimates verschiedener Methoden, welche im Vergleich zur zufälligen Auswahl stehen (Standardabweichungen schattiert gekennzeichnet).

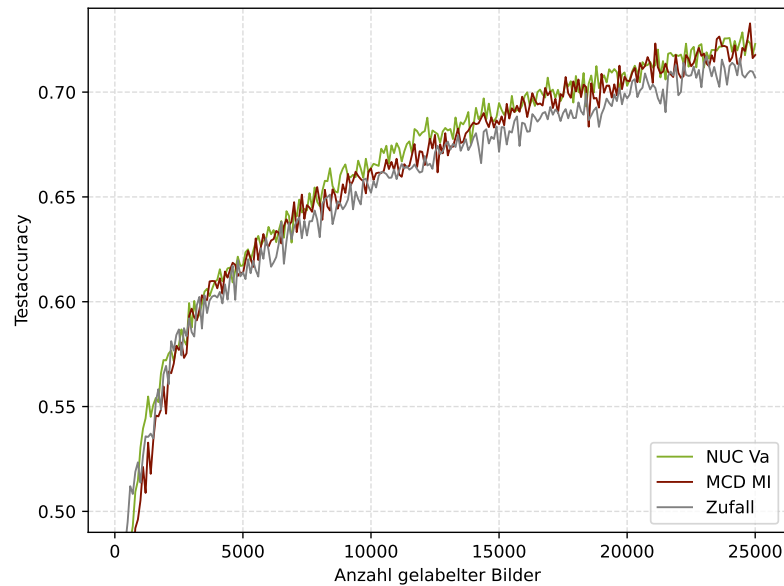


Abbildung 4.6: Entwicklung der Testaccuracy als Funktion der Größe des Trainingsdatensatzes über einen langen Zeitraum der Anwendung des unsicherheitsbasierten aktiven Lernens. Der Trainingsdatensatz wurde stets um 100 zufällige Samples bzw. um die 100 Samples mit den höchsten Uncertainty Estimates anhand der Methoden MCD MI und NUC Va erweitert. Der Prozess wurde einmalig ausgeführt.

eine sehr hohe Testaccuracy erreichen. Folglich werden zufällig ausgewählte Samples häufig schon vor dem erneuten Training richtig klassifiziert, sodass aus ihnen nur wenig neue Informationen über die Datenverteilung zu gewinnen sind. Das sind mögliche Erklärungen für die Beobachtungen von [Gal16]. Bei Cifar10 und dem initialen Trainingsdatensatz L_{01} hingegen hat das Modell zunächst eine Testaccuracy von lediglich 37%. Durch die zufällige Auswahl werden somit viele informative Daten gewählt, die zudem diverser sind als es bei der Auswahl anhand von Unsicherheit der Fall ist.

Abschließend wird die Entwicklung der Testaccuracy über einen langen Zeitraum der Anwendung des aktiven Lernens exemplarisch für zwei der besten Methoden untersucht. Begonnen wird mit einem initialen Trainingsdatensatz von 100 Bildern. Es werden solange 100 Samples hinzugefügt, bis der Trainingsdatensatz L 25000 Bilder beinhaltet. Der Start mit nur wenigen Bildern ist bei Volkswagen am realistischsten, da das aktive Lernen bereits zu Beginn des Modelltrainings verwendet werden soll und häufig nur sehr wenig gelabelte Bilder vorhanden sind. Betrachtet werden die Auswahl zufälliger Samples und die unsicherheitsbasierte Auswahl anhand der Uncertainty Estimates von NUC Va und MCD MI. Der Validierungsdatensatz enthält nun 5000 Bilder. Die Ergebnisse sind in Abbildung 4.6 einsehbar.

Zu Beginn des Trainings zeigt das unsicherheitsbasierte DeepAL erneut kein schnelleres Wachstum der Testaccuracy als die Auswahl zufälliger Samples. Eine Verbesserung des Trainingsprozesses erfolgt erst ab circa 5000 Daten. Hier steigt die Performance mit Uncertainty Estimates der NUC-Methode zunächst schneller als mit Uncertainty Estimates der Mutual Information von MC Dropout. Das unsicherheitsbasierte DeepAL zeigt bei Beginn mit einem kleinen initialen Datensatz folglich erst nach vielen Iterationen Erfolge, was für die Verwendung bei Volkswagen hinderlich sein kann.

4.3 Hybride Auswahlstrategie für das DeepAL

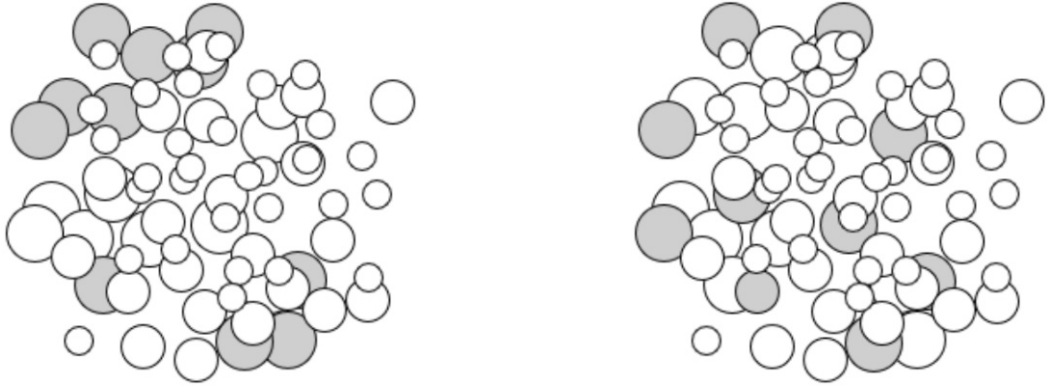
In diesem Abschnitt wird eine Auswahlstrategie betrachtet, die zusätzlich die Diversität der Samples eines Batches berücksichtigt. Die Performance der Strategie wird wie die unsicherheitsbasierte Auswahlstrategie für das aktive Lernen auf dem Cifar10 Datensatz gegenüber der Auswahl zufälliger Samples untersucht.

4.3.1 Ansatz zur Erhöhung der Diversität ausgewählter Samples

Eine Auswahlstrategie, die Samples unabhängig voneinander bewertet und ihren Zusammenhang ignoriert, führt, wie der vorige Abschnitt gezeigt hat, zu unzureichend optimierten Batches. Abbildung 4.7 verdeutlicht die Problematik.

Es gibt verschiedene Ansätze, die die Diversität der Samples eines Batches auswerten. Einer davon ist das von A. Kirsch et al. [KvAG19] entworfene BatchBALD. Dieser Ansatz ermittelt die Joint Mutual Information mehrerer Daten. Die Metrik erfasst ähnlich wie die Mutual Information die prädiktive Unsicherheit eines Samples, aber zusätzlich die Variation in den Prädiktionen aller Samples und somit deren Diversität. Sie ist für bayesche Modelle anwendbar, beispielsweise für ein Netzwerk mit Dropout. Jedoch hat die exakte Berechnung der Joint Mutual Information für große Batches einen viel zu hohen Rechenaufwand, da Matrizen der Größe $c^b \times k$ multipliziert werden. Mit c ist die Anzahl der Klassen und mit b die Batch-Größe gemeint. Im Fall von Monte Carlo Dropout ist k die Anzahl Feedforward Durchläufe mit aktiviertem Dropout. Aus diesem Grund und dem Fakt, dass die Joint Mutual Information nicht für die NUC-Methode anwendbar ist, wird im Folgendem ein anderer Ansatz getestet.

Hybride Auswahlstrategien beachten verschiedene Eigenschaften der Samples und wiegen diese ab [RXC⁺21]. Zum Beispiel erfasst Exploration-P von C. Yin et al. [YQC⁺17] die



(a) Batch einer Auswahlstrategie, die nur die Informationsgröße berücksichtigt. (b) Auswahlstrategie, die Informationsgröße und Diversität des Batches beachtet.

Abbildung 4.7: Vergleichende Visualisierung der ausgewählten Batches von zwei verschiedenen Bewertungsstrategien. Eine Strategie beachtet nur die Höhe des individuellen Informationsgehalts der Samples für das Modelltraining und die andere bewertet Informationsgröße sowie die Diversität der Informationen. Ein Kreis repräsentiert jeweils ein Sample, die Größe der Kreise steht für die Information und die Nähe zu anderen Kreisen indiziert Ähnlichkeit der Samples (Grafik von [RXC⁺21]).

Unsicherheit und Redundanz. Feature Repräsentationen der Samples werden verwendet, um die Ähnlichkeit zweier Samples zu berechnen, während die Unsicherheit der Samples separat abgeschätzt wird. Exploration-P sieht die Verwendung der Softmaxentropie vor, um die prädiktive Unsicherheit abzuschätzen. Die Auswahlstrategie ist allerdings leicht auf andere Uncertainty Estimation Methoden anpassbar. Die Unsicherheit eines ausgewählten Samplebatches S sei $E(S) = \sum_{\mathbf{x}_i \in S} \text{uncert}(f(\mathbf{x}_i))$ und die Ähnlichkeit ist definiert als

$$R(S) = \sum_{\mathbf{x}_i \in S} \sum_{\mathbf{x}_j \in S} \text{Sim}(\mathbf{x}_i, \mathbf{x}_j), \quad \text{Sim}(\mathbf{x}_i, \mathbf{x}_j) = f'(\mathbf{x}_i) \mathcal{M} f'(\mathbf{x}_j), \quad (4.3)$$

wobei $f'(\mathbf{x})$ die extrahierten Features eines Samples \mathbf{x} aus dem Netzwerk f bezeichnet und $\text{Sim}(\mathbf{x}_i, \mathbf{x}_j)$ die Ähnlichkeit zweier Samples misst. \mathcal{M} ist eine Ähnlichkeitsmatrix. Falls \mathcal{M} die Einheitsmatrix ist, ist $\text{Sim}(\mathbf{x}_i, \mathbf{x}_j)$ das Produkt der Featurevektoren. Die Bewertungsfunktion setzt sich aus beiden Kriterien zusammen:

$$Q_f(S) = E(S) - \frac{\alpha}{|S|} R(S). \quad (4.4)$$

Der Parameter α wird verwendet, um das Gewicht der beiden Eigenschaften Unsicherheit und Gleichheit zu bestimmen. Um bei der nachfolgenden Untersuchung auf die Anpassung des Parameters zu verzichten, fiel der Entschluss, die zwei Eigenschaften multi-kriteriell zu betrachten. Die Lösung eines multi-kriteriellen Problems ist eine Menge mehrerer optimaler Lösungen. Sie wird als Pareto Front bezeichnet. Sei $\mathbf{l}(\theta_1) \preceq \mathbf{l}(\theta_2)$, falls für alle Kriterien l_i gilt: $l_i(\theta_1) \leq l_i(\theta_2)$. Der Fall $\mathbf{l}(\theta_1) \prec \mathbf{l}(\theta_2)$ liegt vor, wenn $\mathbf{l}(\theta_1) \preceq \mathbf{l}(\theta_2)$ und es ein Kriterium l_i gibt, sodass $l_i(\theta_1) < l_i(\theta_2)$ ist. Ein Datenpunkt θ_1 dominiert θ_2 falls $\mathbf{l}(\theta_1) \prec \mathbf{l}(\theta_2)$ gilt [NSFC21]. Hierbei ist \prec nicht zwangsläufig als *kleiner als* zu betrachten, sondern vielmehr als *optimaler als*. Ein Datenpunkt befindet sich in der Pareto Front, wenn er von keinem anderen dominiert wird. Die Kriterien für die hybride Auswahlstrategie des DeepAL sind $E(S)$ und $R(S)$.

4.3.2 Anwendung der hybriden Auswahlstrategie für Cifar10

Die soeben beschriebene, abgewandelte Form der Interpolation-P Auswahlstrategie soll nun für die in Abschnitt 4.2 entworfene Testumgebung des aktiven Lernens angewandt werden. Ziel ist es, herauszufinden, ob die Performance der Uncertainty Estimation Methoden, die im vorangegangenen Abschnitt die besten Ergebnisse lieferten, durch eine hybride Auswahlstrategie verbessert werden kann. Dies sind die Methoden NUC, trainiert auf Validierungsdaten (NUC Va) sowie die Mutual Information des MC Dropouts (MCD MI) und des Data Augmentation Ensembles (DA MI).

Um Rechenoperationen zu sparen, wird der Repräsentationsvektor $r(\mathbf{x})$ einer der letzten Schichten des Netzwerks als $f'(\mathbf{x})$ und die Einheitsmatrix für \mathcal{M} verwendet. Die letzten Schichten eines Netzwerks geben meistens eine sehr komprimierte Repräsentation der Eingabe aus, sodass der Vektor $r(\mathbf{x})$ möglichst wenige Einträge hat. Außerdem wird die Softmaxfunktion $f'(\mathbf{x}) = \text{softmax}(r(\mathbf{x}))$ auf den Repräsentationsvektor angewandt. Somit werden zwei Samples \mathbf{x}_i und \mathbf{x}_j , deren Repräsentationsvektoren nur kleine Einträge haben, nicht aufgrund des geringen Produkts $r(\mathbf{x}_i)\mathcal{M}r(\mathbf{x}_j)$ fälschlicherweise mit einer geringen Ähnlichkeit assoziiert. Durch die Softmaxfunktion ist die Summe aller Einträge von $f'(\mathbf{x})$ stets 1.

Da die Bewertung jedes möglichen Batches S viel zu aufwendig ist, wird für diesen Test nur ein Ausschnitt des Suchraums betrachtet, indem alle Bilder aus dem Datenpool U zufällig in Batches aufteilt werden. Es folgt die zehnfache Wiederholung einer solchen Aufteilung, sodass jedes Sample in 10 Batches vorkommt. Erneut werden die initialen Trainingsdatensätze L_{01} , L_{02} und L_{03} mit $|L_{01}| = 100$, $|L_{02}| = 1000$ und $|L_{03}| = 10000$ verwendet und es wird zwischen den Batch-Größen 100 und 1000 unterschieden. Der Aufbau der Tests ist exakt derselbe wie im Abschnitt 4.2, damit die Ergebnisse vergleichbar sind.

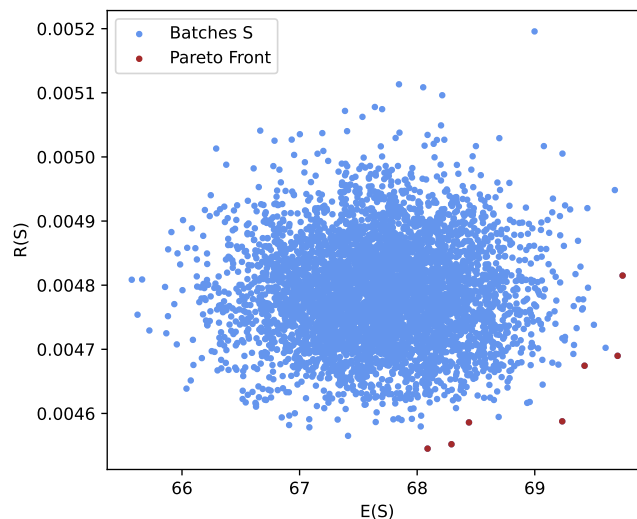


Abbildung 4.8: Beispielhafte Darstellung der Ähnlichkeit $R(S)$ und der Unsicherheit $E(S)$ der Methode NUC Va für jeden betrachteten Batch S bestehend aus 100 Samples. Die Pareto Front ist farblich hervorgehoben. Da $E(S)$ zu maximieren und $R(S)$ zu minimieren ist, befindet sie sich im rechten, unteren Teil der Punktmenge.

In jeder Iteration erfolgt die zufällige Wahl eines Batches S der Pareto Front. Für eine erste Iteration mit Uncertainty Estimates von NUC Va ist die Pareto Front in Abbildung 4.8 veranschaulicht.

Die Ergebnisse nach hybrider Auswahl neuer Daten sind in Abbildung 4.9 dargestellt. Lediglich beim Start mit 100 Bildern im Trainingsdatensatz kann eine etwas bessere Performance der hybriden Auswahl im Vergleich zur zufälligen Auswahl erahnt werden. Die unsicherheitsbasierte Auswahl im vorherigen Abschnitt 4.2 war in diesem Fall nicht besser als die zufällige, die hybride Auswahl mit der Mutual Information des Data Augmentation Ensembles hingegen schon. Bei den anderen drei Testfällen ist jedoch keine Verbesserung gegenüber der Wahl zufälliger Samples zu vermerken.

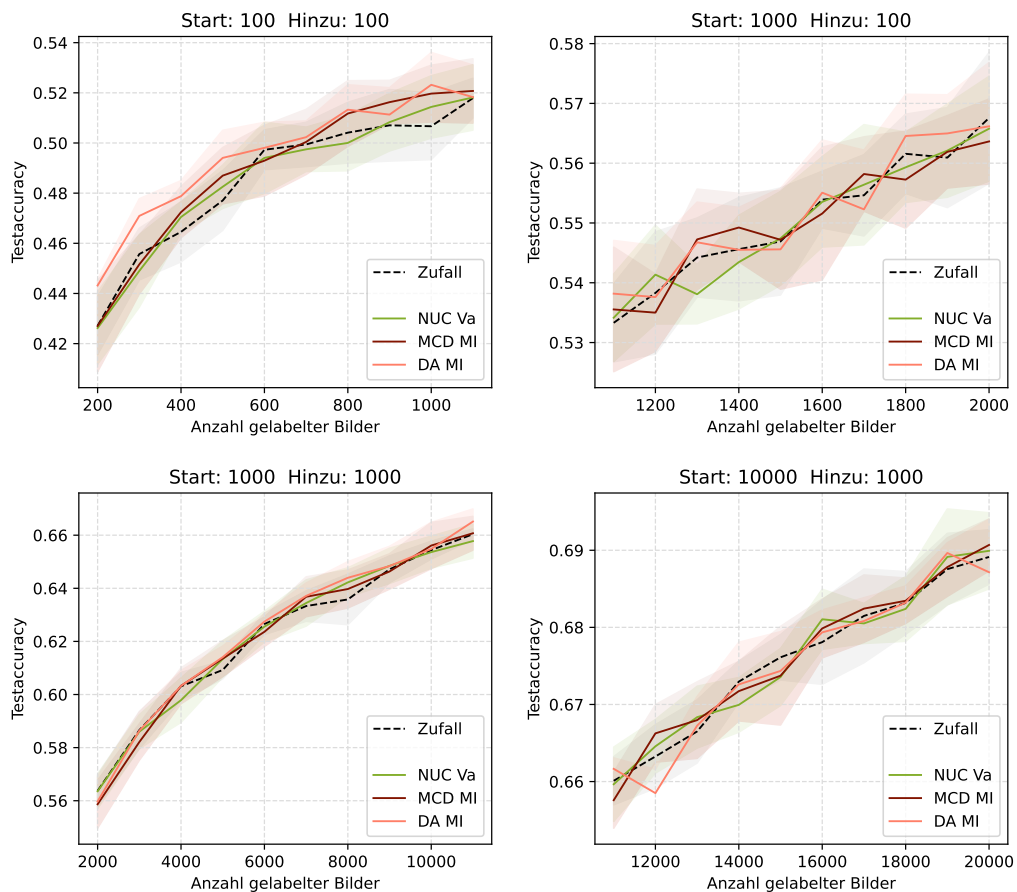


Abbildung 4.9: Wachstum der Testaccuracy als Funktion der Größe des Testdatensatzes für die hybride Auswahlstrategie. Unterschieden wird zwischen verschiedenen großen initialen Datensätzen und unterschiedlich großen Mengen an Samples, die dem Trainingsdatensatz pro Iteration hinzugefügt werden. Dargestellt sind die Mittelwerte nach fünffacher Wiederholung der Simulationen. Standardabweichungen sind schattiert gekennzeichnet.

Dass die Testaccuracy für die hybriden Auswahlstrategien kaum schneller steigt als durch zufällige Samples, kann damit zusammenhängen, dass nur ein sehr kleiner Bereich aller möglichen Batches S betrachtet wird. Folglich wird vermutlich stets ein Batch zu den Trainingsdaten hinzugefügt, der noch weit vom Optimum entfernt ist. Lediglich 10 verschiedene Aufteilungen der ungelabelten Samples in Batches wurden betrachtet. Beim Start mit 100 Trainingsdaten und 100 Validierungsdaten sind 49800 ungelabelte Daten

im Pool U . Somit erfolgten die Berechnungen von $E(S)$ und $R(S)$ bei der Batch-Größe 100 für 4980 verschiedene Batches. Insgesamt gibt es jedoch $\binom{49800}{100}$ verschiedene Batches, was derart viele sind, dass die Anzahl nicht berechenbar ist.

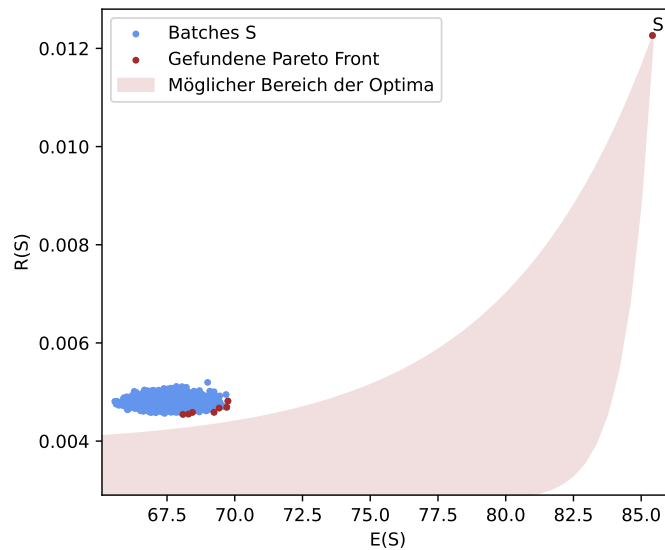


Abbildung 4.10: Darstellung der Ähnlichkeit $R(S)$ und Unsicherheit $E(S)$ der Methode NUC Va für zufällig zusammengestellte Batches S und für den Batch S' bestehend aus den Samples mit den höchsten Uncertainty Estimates. Die Batches enthalten jeweils 100 Samples und die Pareto Front ist farblich hervorgehoben. Aufgrund der hohen Distanz von $(E(S'), R(S'))$ zu den anderen Punkten ist zu vermuten, dass die gefundene Pareto Front noch weit von der tatsächlichen entfernt ist.

Abbildung 4.10 ist ein Anzeichen dafür, dass der ausgewählte Batch noch weit vom optimalen Batch entfernt ist. Hier sind die Unsicherheit E der Methode NUC Va und die Ähnlichkeit R verschiedener Batches visualisiert. Betrachtet werden die zufällig zusammengestellten Batches und der Batch S' , der nur aus den Samples mit den höchsten Uncertainty Estimates besteht. S' wäre von der unsicherheitsbasierten Auswahlstrategie in Abschnitt 4.2 ausgewählt worden. Die Darstellung bestätigt einerseits, dass die Samples mit der höchsten prädiktiven Unsicherheit tatsächlich sehr ähnlich sind. $R(S')$ ist deutlich größer als die durchschnittliche $R(S)$, was die Verwendung einer hybriden Auswahlstrategie motiviert. Andererseits ist die große Distanz des Punktes $(E(S'), R(S'))$ zu den Punkten, die in diesem Abschnitt für die hybride Strategie betrachtet wurden, ein Anzeichen dafür, dass noch andere, besser geeignete Batches weit entfernt liegen können. Die mögliche Fläche, in der sich die tatsächliche Pareto Front vermutlich befindet, ist in Abbildung 4.10 schattiert gekennzeichnet. Bekannt ist lediglich, dass der Batch S' zu den optimalen Lösungen gehört, da $E(S')$ optimal ist. Folglich, kann $(E(S'), R(S'))$ von keinem anderen Punkt dominiert werden.

Es kann geschlussfolgert werden, dass die Betrachtung einzelner zufälliger Samplemengen für die hybride Auswahlstrategie nicht zielführend ist. Jedoch steigt der Rechenaufwand mit jeder Bewertung eines weiteren Batches. Wichtig für die Anwendung einer hybriden Auswahlstrategie ist somit ein effizienter Optimierungsalgorithmus für die Suche nach einem (fast) idealen Batch S .

5 Fazit und Ausblick

In diesem Kapitel werden die wichtigsten Erkenntnisse der Arbeit zusammengefasst und bezüglich der Zielsetzung der Arbeit diskutiert. Darüber hinaus wird ein Einblick in weitere aktuelle sowie zukünftig notwendige Entwicklungen der Uncertainty Estimation und des aktiven Lernens gegeben. Die betrachteten Methoden zur Uncertainty Estimation sind neben der Klassifizierung auch für komplexere Probleme verwendbar. Eine Anwendung mit großen Parallelen zur Klassifizierung ist die semantische Segmentierung, die in diesem Kapitel genauer betrachtet wird. Dabei wird auf notwendige Anpassungen der Methoden und Vorgehensweisen dieser Arbeit sowie auf aktuelle Ergebnisse der Forschung eingegangen. Anschließend werden allgemeine Problematiken der Uncertainty Estimation erläutert und geklärt, wie diese in Zukunft vermindert werden können. Zum Schluss wird ein Ausblick für das Deep Active Learning (DeepAL) gegeben und ein Fazit für dessen Verwendung bei Volkswagen gezogen.

5.1 Zusammenfassung und Auswertung der Erkenntnisse

Zunächst ist zu erwähnen, dass die Feststellung, welche Methode am besten für die Uncertainty Estimation geeignet ist, schwierig ist. Die einzelnen Methoden haben oft bei verschiedenen Tests Vorteile und sogar unterschiedliche Datensätze und Modellarchitekturen führen zu anderen Resultaten beim Vergleich der Methoden.

Die Kalibrierung von Uncertainty Estimates kann mit Techniken wie der isotonischen Regression verbessert werden, sodass die Ausgaben von einem Menschen interpretierbar sind. Die Kalibrierung repräsentiert somit weniger die eigentliche Qualität einer Methode zur Uncertainty Estimation. Für die Erkennung korrekter und inkorrekt Prädiktionen ist der NUC am besten geeignet, insofern genug Daten für das Training des externen Netzwerks vorhanden sind (mindestens 10000 Trainingsdaten bei Cifar10). Die beiden Versionen für das Training des externen Netzwerkes unterscheiden sich hinsichtlich weniger Kriterien. Die Certainty Scores des auf den Validierungsdaten trainierten Netzwerks sind bereits ohne Anwendung der isotonischen Regression gut kalibriert und haben eine höhere Qualität, falls Overfitting vorliegt. Die Qualität des auf den Trainingsdaten trainierten Netzwerks ist hingegen weniger von der Anzahl betrachteter Nearest Neighbors (k) abhängig. Für Deep Neural Networks (DNNs), die mit wenig Daten trainiert werden, erreichen die Uncertainty Estimates der Softmaxentropie der Netzwerkausgabe sowie die prädiktiven Entropien von Ensemble Methoden und dem MC Dropout die besten Ergebnisse. Für die Out-of-Domain (OOD) Erkennung eignen sich die prädiktiven Entropien ebenfalls sehr, die Softmaxentropie hingegen deutlich weniger. Auch die Qualität des NUCs variiert stark für die OOD Erkennung bei verschiedenen DNNs. Bezüglich der Laufzeit, gehört der NUC für komplexe Modelle zu den schnellsten Methoden. Schneller ist nur die Softmaxentropie. Die Ensemble Methoden haben den großen Nachteil einer sehr langen Vorbereitungszeit durch

das Training des Ensembles. Das MC Dropout kann nur auf DNNs mit Dropoutschichten angewandt werden und die Berechnung eines Uncertainty Estimates dauert vergleichsweise lange, vor allem bei komplexen Modellen.

Anhand dieser Testergebnisse ist festzustellen, dass sich der NUC für DNNs, die mit großen Datensätzen trainiert werden, am besten eignet und bei kleinen Modellen bereits die einfache Softmaxentropie ausreicht, vorausgesetzt, dass bei der konkreten Anwendung keine OOD Daten auftreten. Falls es vorkommen kann, dass zur Testzeit OOD Samples vorliegen, so ist die prädiktive Entropie von Ensembles oder dem MC Dropout zu bevorzugen.

Für das Deep Active Learning (DeepAL) sind Uncertainty Estimates der Methoden NUC und der Mutual Information der Ensembles oder des MC Dropouts am besten geeignet. Methoden und Metriken, die hauptsächlich oder sogar ausschließlich die epistemische Unsicherheit modellieren, sollten verwendet werden. Konkreter erreichten bei den Simulationen des Kapitels 4 die auf Validierungsdaten trainierte NUC-Methode, die Mutual Information des MC Dropouts und die des Data Augmentation Ensembles die besten Ergebnisse. Da die Simulationen lediglich für eine Netzwerkarchitektur und einen Datensatz ausgeführt worden sind, muss dies nicht immer gelten. In Anbetracht des hohen Rechenaufwands für das Training der Ensemblemitglieder nach jeder Iteration des DeepAL sind die Ensembles vergleichsweise weniger effizient. Das MC Dropout hat den Nachteil, dass die Laufzeit im Vergleich zum NUC mit wachsender Komplexität des Netzwerks und wachsender Menge ungelabelter Daten deutlich schneller steigt. Folglich ist die deterministische Uncertainty Estimation Methode, der Neighborhood Uncertainty Classifier, am ehesten für das aktive Lernen zu empfehlen.

Eine wichtige Beobachtung für das aktive Lernen ist, dass die ausschließliche Betrachtung der Unsicherheit keine optimale Auswahlstrategie ist. Bei einem großen Datenpool U und einem kleinen initialen Trainingsdatensatz L_{01} ist diese Auswahlstrategie oft schlechter als die zufällige Auswahl neuer Trainingsdaten. Es werden zu viele ähnliche Samples ausgewählt, die anschließend den Trainingsdatensatz dominieren. Mittels hybrider Auswahlstrategien ist es möglich zusätzlich die Diversität der Samples in einem Batch zu berücksichtigen. Jedoch gibt es sehr viele verschiedene Batches, sodass die Suche nach einem geeigneten Batch sehr aufwendig ist.

Die Frage, ob die häufig verwendeten Qualitätstests der Uncertainty Estimation aus Kapitel 3 repräsentativ für die Performance der Methoden beim aktiven Lernen sind, ist zu verneinen. Die Kalibrierung einer Methode ist für das DeepAL irrelevant. Benötigt wird lediglich die Ordnung der Samples nach ihrer prädiktiven Unsicherheit. Dabei ist es egal, ob die Werte im Intervall $(0, 1)$ liegen. Auch die Ergebnisse der Performancemetriken AUROC und AUPR sind weit von denen des aktiven Lernens entfernt. In Kapitel 4 wird gezeigt, dass für das aktive Lernen nur die epistemische Unsicherheit relevant ist. Die meisten Tests für die Uncertainty Estimation sind jedoch so designt, dass Methoden und Metriken, die die komplette prädiktive Unsicherheit erfassen, die besten Ergebnisse erzielen. Die prädiktiven Entropien und die Softmaxentropie haben für das CNN c10₁₀₀, das CNN c10₁₀₀₀ und das CNN c10₁₀₀₀₀ die höchsten AUROCs und AUPRs, sind jedoch am schlechtesten für das aktive Lernen geeignet. Dass Uncertainty Estimation Methoden gut OOD Daten erkennen, kann nützlich sein, um Fehler in der Datenbeschaffung zu erkennen, ist jedoch für den Prozess des aktiven Lernens nicht wichtig. Da OOD Daten

eine hohe epistemische Unsicherheit haben, würden sie im Idealfall von der unsicherheitsbasierten Auswahlstrategie ausgewählt werden und der Mensch würde beim Labeln feststellen, dass OOD Daten fehlerhaft in den Datenpool U gelangt sind. Die Laufzeit der Uncertainty Estimation Methoden ist sehr wichtig für das DeepAL, da das Ziel ist, den Trainingsprozess neuronaler Netzwerke zu optimieren. Sind die Techniken des DeepALs mit zusätzlichen Kosten verbunden, so ist das Training trotz verringertem Labelaufwand nicht unbedingt effizienter. Folglich ist die Laufzeitmessung der einzige Test aus Kapitel 3 von hoher Relevanz für das aktive Lernen.

5.2 Uncertainty Estimation für semantische Segmentierung

Die semantische Segmentierung bezieht sich wie die Klassifizierung auf Bilddaten. Ziel ist, diese in Bereiche bestimmter Kategorien aufzuteilen, wie es im jeweils zweiten Bild von links in Abbildung 5.1 der Fall ist. Jedem Pixel wird eine Klasse zugeordnet, weshalb die semantische Segmentierung als Klassifizierung jedes einzelnen Pixels eines Bildes interpretiert werden kann. DNNs geben Softmaxscores für die Zugehörigkeit jedes einzelnen Pixels bezüglich jeder Klasse aus.

Mit dem MC Dropout und den Ensemble Methoden erhält man für ein Eingabebild \mathbf{x} erneut mehrere verschiedene Ausgaben der Feedforward Läufe und kann so die prädiktive Wahrscheinlichkeitsverteilung ermitteln. Für die simple Klassifizierung werden Uncertainty Estimates mit den Metriken prädiktive Entropie und Mutual Information ermittelt (vergleiche Kapitel 3.3.2). Ein intuitiver Weg, diese Methoden hier zu nutzen, ist die Anwendung der prädiktiven Entropie und Mutual Information auf die Netzwerkausgaben pro Pixel. Folglich sind hier im Vergleich zur simplen Klassifizierung kaum Anpassungen von Nöten. Anzumerken ist, dass die Softmaxentropie für die semantische Segmentierung ebenfalls direkt auf die einzelnen Softmaxausgaben des Netzwerks angewandt werden kann und somit als Maß für die aleatorische Unsicherheit dient. Die Mutual Information repräsentiert analog zur simplen Klassifizierung die epistemische und die prädiktive Entropie sowohl epistemische als auch aleatorische Unsicherheit. [MvATG21, GW19]

Um den hohen Rechenaufwand, der beim MC Dropout und den Ensemble Methoden allein durch die vielen Feedforward Durchläufe entsteht, zu vermeiden, konzentrieren sich viele Forscher und Forscherinnen auf die Entwicklung effizienterer Methoden. Für das autonome Fahren, einer Anwendung bei der die semantische Segmentierung sehr bedeutsam ist, sind Methoden mit hohem Rechen- und Zeitaufwand unpraktisch. C. J. Holder et al. [HS21] trainieren ein einzelnes, kompaktes Modell darauf, die Ausgaben eines großen Ensembles nachzuahmen. J. Mukhoti et al. [MvATG21] erweiterten eine deterministische Methode, die die Dichte im Feature Raum zur Uncertainty Estimation verwendet, für die semantische Segmentierung. Mittels eines einzigen Feedforward Laufs durch das Netzwerk werden hier Uncertainty Estimates für jedes Pixel erhalten. Der für die Klassifizierung untersuchte Neighborhood Uncertainty Classifier ist ebenfalls eine deterministische Methode. Der NUC verwendet ein externes Netzwerk und ermittelt Certainty Estimates für ein Hauptmodell anhand der Dichte in dessen Repräsentationsraum [RM19]. Eine Anpassung

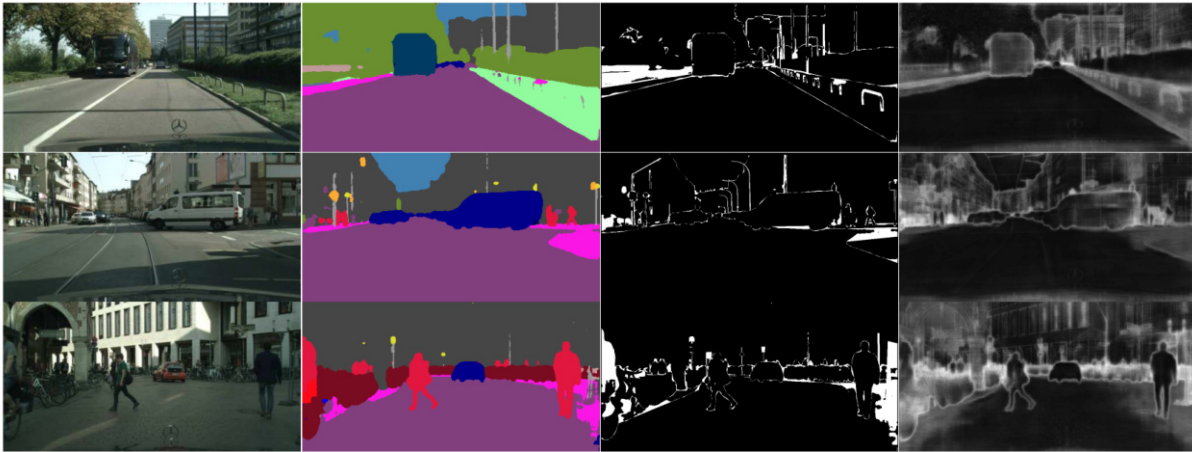


Abbildung 5.1: Bilder aus dem Cityscapes Testdatensatz mit (von links nach rechts) der Ausgabe eines Segmentierungsmodells, einer Darstellung korrekt klassifizierter Pixel und der Visualisierung der pixelweisen Unsicherheit. Weiße Pixel in der dritten Darstellung von links wurden inkorrekt klassifiziert. Für die Unsicherheitsvisualisierung gilt, je heller ein Pixel, desto höher ist das Uncertainty Estimate. (Abbildung von [HS21])

dieser für die simple Klassifizierung eingeführten Methode an die semantische Segmentierung kann ebenfalls möglich sein. Das Training eines externen Netzwerks für jedes einzelne Pixel wäre sehr ineffizient. Vielmehr soll ein Netzwerk die Certainty jedes Pixels abschätzen. Das externe Netzwerk benötigt in diesem Fall für jedes Pixel ein Neuron in der Ausgabeschicht, das darauf trainiert wird, vorherzusagen, ob das Hauptmodell dieses Pixel korrekt klassifiziert hat. Diese angepasste Methode würde neben dem Feedforward Lauf des Hauptmodells nur einen weiteren Lauf durch das sehr kompakte externe Netzwerk benötigen, um die pixelweisen Unsicherheiten abzuschätzen. Die Umsetzung einer solchen Anpassung hat somit Potenzial für zukünftige Arbeiten.

Die pixelweisen Uncertainty Estimates ermöglichen eine Visualisierung der Verteilung der Unsicherheit über das Eingabebild. C. J. Holder et al. [HS21] repräsentieren ihre Uncertainty Estimates beispielhaft für drei Bilder des Cityscapes Datensatzes in einer solchen Darstellung, wie in Abbildung 5.1 zu sehen ist. Der Cityscapes Datensatz enthält Aufnahmen von Straßenszenen verschiedener Städte mit hoher Auflösung, der die Evaluierung pixelweiser Ausgaben von Segmentierungsmodellen ermöglicht [COR⁺16].

Bewertungen der Uncertainty Estimates für die Pixel können wie für die Uncertainty Estimates bei der Klassifizierung in Kapitel 3 erfolgen. Pixel mit ähnlicher Unsicherheit werden gruppiert und deren durchschnittliche Korrektheit bestimmt. Somit können Kalibrierungsfehler wie der ECE ermittelt werden [CWo⁺21]. Anhand der Korrektheit der Klassifizierung und des Uncertainty Estimates des Pixels ist es möglich, ebenfalls Metriken wie Precision, Recall, Specificity, AUROC und AUPR auszuwerten.

Das aktive Lernen kann auch für die semantische Segmentierung verwendet werden und eine Performanceverbesserung bewirken. Das DeepAL ist in diesem Bereich noch deutlich weniger untersucht worden als für die Klassifizierung [GK20], obwohl der Labelaufwand bei semantischer Segmentierung viel größer ist. Jeder Pixel muss annotiert werden, weshalb hier in der Reduzierung des Labelaufwands ein besonders hoher Nutzen zu erwarten

ist. Aufgrund des hohen Labelaufwands sollten für die semantische Segmentierung kleinere Batch-Größen verwendet werden, als in Kapitel 4 für die Klassifizierung betrachtet wurden. Die pixelweisen Uncertainty Estimates werfen für das unsicherheitsbasierte aktive Lernen die Frage auf, wie entschieden werden soll, welche Bilder gelabelt werden. Während Bilder bei der Klassifizierung eindeutig nach ihrer Unsicherheit bewertet werden können, gibt es bei der semantischen Segmentierung nicht einen Wert, der die Unsicherheit der gesamten Ausgabe widerspiegelt. Hier können verschiedene Vorgehensweisen betrachtet werden. Einerseits ermöglicht die durchschnittliche oder maximale pixelweise Unsicherheit eine Bewertung des gesamten Bildes. Andererseits ist es möglich, lediglich vereinzelte Pixel mit hohem Uncertainty Estimate für das Labeling auszuwählen. Hier muss entschieden werden, ab welcher Höhe der Unsicherheit ein Pixel für das Labeling ausgewählt wird oder ob stets ein fester Prozentsatz des Bildes gelabelt wird. Außerdem können sich die Kosten für das Labeling bestimmter Bildregionen unterscheiden [GK20]. Auch das Pseudo-Labeling von Pixeln mit geringer Unsicherheit, indem die Prädiktion als Label verwendet wird, ist möglich [GiNCF17].

Abschließend ist festzustellen, dass der Aufwand für die Anpassung der für Klassifizierung verwendeten Uncertainty Estimation je nach Methode unterschiedlich hoch ist. Die prädiktive Entropie und die Mutual Information vom MC Dropout und den Ensembles können fast unverändert für semantische Segmentierung angewandt werden. Am groben Ablauf des unsicherheitsbasierten DeepAL muss für die semantische Segmentierung ebenfalls nur wenig verändert werden. Jedoch gibt es bei der Auswahlstrategie allein anhand der Unsicherheit als einzelner Kriterium viele Variationen.

5.3 Aktuelle Probleme und nötige Verbesserungen der Uncertainty Estimation

Trotz der vielen Fortschritte, die bei der Quantifizierung prädiktiver Unsicherheiten in den letzten Jahren erreicht wurden, ist die Anwendung der Uncertainty Estimation in praktischen Anwendungen begrenzt. Gründe hierfür werden im Folgendem erläutert.

Für viele reale Probleme fehlen angemessene Validierungen der Uncertainty Estimates. Datensätze wie Cifar10, Cifar100 und ImageNet sind wenig spezifisch und Resultate auf diesen Datensätzen daher schwer auf komplexe, echte Umgebungen anwendbar. Beispielsweise sind Daten eines Satelliten mit geringer Auflösung stark durch Rauschen beeinflusst. Außerdem basieren viele Evaluierungsmetriken auf dem gesamten Testdatensatz, so zum Beispiel der Expected Calibration Error (ECE). Die Auswertungen erfolgen empirisch, denn exakte Zielwerte für die prädiktive Unsicherheit sind im Allgemeinen nicht vorhanden. Dadurch wird die Performance der Unsicherheit, die für einzelne Samples abgeschätzt wird, stark von der Performance auf dem Rest der Daten beeinflusst. Für praktische Anwendungen würde der Zugriff auf die Zuverlässigkeit eines einzelnen Uncertainty Estimates deutlich mehr Möglichkeiten bieten als die durchschnittliche Zuverlässigkeit für die Testdaten, welche von der aktuellen Situation unabhängig sind. Zudem unterscheiden häufig verwendete Evaluierungsmetriken wie die AUROC und die AUPR nicht zwischen aleatorischer und epistemischer Unsicherheit. Dies ist jedoch für die Bewertung einer Uncertainty Estimation Methode für Anwendungen wie das aktive

Lernen, bei denen nur eine Art der prädiktiven Unsicherheit interessant ist, wichtig. [KF14, GTA⁺22]

Ein weiteres Problem ist die Tatsache, dass Methoden, die für einen bestimmten Datensatz kalibriert wurden, aufgrund der Domänenverschiebung nicht ohne Anpassung der Kalibrierung auf andere Daten angewandt werden können. Kapitel 3.3.1 hat bereits gezeigt, dass bei Anwendung der isotonischen Regression zur Kalibrierung von Metriken wie den Entropien und der Mutual Information unterschiedliche Regressionsfunktionen je nach Ensemble Methode und Datensatz berechnet werden. Dass der Wertebereich dieser Metriken von der Anzahl der Klassen abhängig ist, verstärkt den Unterschied zwischen den Funktionsgraphen. Um die Kalibrierung angemessen anpassen zu können, sind viele Daten nötig. Oftmals werden synthetische Daten verwendet, da das Aufbereiten und Labeln echter Daten sehr kostspielig ist. Die tatsächliche Kalibrierungsqualität auf den echten Daten kann jedoch schlechter sein. [GTA⁺22]

Die Bereitstellung eines Evaluierungsprotokoll mit grundlegenden Datensätzen und konkreten Evaluierungsmetriken würde sicherlich den Forschungsfortschritt in Zukunft beschleunigen. Zusätzlich könnten Experten von Domänen realer Anwendungen Vergleiche verschiedener Ansätze durchführen und die Schwächen bestimmter Methoden in ihrer Domäne präsentieren. Um einzelne Uncertainty Estimates bewerten zu können, sind exakte Zielwerte besonders wichtig. Dies ist aktuell jedoch ein schwer zu lösendes Problem. Vermutlich müssten hierfür alle möglichen Quellen der Unsicherheit vom Prozess des Labelings bis zur Anwendung des fertig trainierten Modells genauestens untersucht werden. [GTA⁺22]

Hinsichtlich neuronaler Netzwerke ist die mangelnde Nachvollziehbarkeit der Entscheidungen ein häufiger Kritikpunkt. Die Transparenz der Berechnung ist nicht gegeben. Dies führt zu mangelndem Vertrauen der Anwender in die Ausgaben des Netzwerks. Uncertainty Estimates sind ein Anzeichen dafür, ob der Prädiktion vertraut werden kann. Allerdings sind Uncertainty Estimates selbst nicht zu 100% korrekt und wenig nachvollziehbar, da die Methoden zur Uncertainty Estimation keine Begründung geben, weshalb die vorliegenden Prädiktionen sicher ist. Daher ist es besonders wichtig, zu verstehen wie die Methoden funktionieren und wie sie die prädiktive Unsicherheit ermitteln. Das Erklären des Verhaltens der Methoden ist ein großer Bereich in der Forschung und noch sehr schwierig detailliert zu erfassen. Eine erste Methode zur Erklärung von Uncertainty Estimates stellten J. Antorán et al. [ABA⁺21] vor. Nachvollziehbare Uncertainty Estimates würden nicht nur die Entwicklung und Verbesserung von Methoden erleichtern, sondern auch ein tiefgründigeres Verständnis des Entscheidungsprozesses neuronaler Netzwerke ermöglichen. [GTA⁺22]

Eine sehr große Gefahr für die Uncertainty Estimation neuronaler Netzwerke geht von sogenannten Adversarial Angriffen aus. Typische Adversarial Angriffe erreichen durch gezielte, für Menschen nicht sichtbare Veränderungen der Eingaben, dass das Modell für diese Eingaben inkorrekte Prädiktionen berechnet. I. Galil et al. [GEY21] gelang es mit einem ähnlichen Angriffskonzept anstatt der Accuracy, die Uncertainty Estimates zu manipulieren. Sie veränderten korrekt klassifizierte Eingaben derart, dass sie sich einer Entscheidungsgrenze (Decision Boundary) des Modells nähern. Den Uncertainty Estimation Methoden wird dadurch eine hohe aleatorische Unsicherheit vorgespielt, sodass korrekten Prädiktionen eine größere Unsicherheit vorhergesagt wird als den tatsächlich inkorrekten Prädiktionen. Die Arbeit von H. Zeng et al. [ZYZ⁺22] führt die Uncertainty

Estimation für Out-of-Domain (OOD) Daten in die Irre. Sie zeigten, dass selbst die besten Uncertainty Estimation Methoden durch den angewandten Adversarial Angriff sehr hohe Certainty Estimates für die OOD Samples ausgeben. Bei Anwendungen wie dem autonomen Fahren, medizinischen Diagnosen oder digitalen Finanzsystemen mit echten, schwerwiegenden Konsequenzen kann die Anfälligkeit der Uncertainty Estimation die Sicherheit und das Leben der Menschen bedrohen. Die Entwicklung robuster Algorithmen und qualitativer Abwehrmechanismen für solche Angriffe ist dringend notwendig, um in Zukunft Uncertainty Estimation häufiger in praktische Anwendungen zu integrieren. [ZYZ⁺22, GEY21]

5.4 Ausblick des DeepALs und dessen zukünftige Verwendung bei Volkswagen

Der aktuelle Forschungsschwerpunkt des DeepAL liegt unter anderem auf der Verbesserung der Auswahlstrategien. Strategien, die die Unsicherheit oder die Diversität betrachten, sind im Fokus. Dabei werden hybride Auswahlstrategien immer häufiger favorisiert [RXC⁺21].

Um den hohen kombinatorischen Aufwand beim Bewerten jedes möglichen Batches zu vermeiden, sind effiziente Algorithmen, die möglichst schnell eine geeignete Samplemenge finden, von Nöten. Beispielsweise haben A. Kirsch et al. [KvAG19] einen Algorithmus vorgestellt, der, beginnend mit einem leeren Batch, iterativ das am besten geeignete Sample hinzufügt, bis die gewünschte Batch-Größe erreicht ist. Hierbei wird jede Samplemenge, bestehend aus den Samples des aktuellen Batches und einem neuen Sample, anhand der Auswahlkriterien bewertet. A. Kirsch et al. [KvAG19] konnten zeigen, dass der Algorithmus $1 - \frac{1}{e}$ -approximiert ist. Das bedeutet, der gefundene Batch ist mindestens $1 - \frac{1}{e} \approx 0.6321$ Mal so gut ist wie der optimale. Bei einer Batch-Größe von b und $|U|$ ungelabelten Samples werden $b \cdot |U|$ Iterationen durchgeführt, was für eine große Menge ungelabelter Samples U und ein großes b noch immer mit einem hohen Rechenaufwand verbunden ist. Im Rahmen dieser Bachelorarbeit wurde versucht, den Algorithmus zu nutzen. Jedoch wies die Auswahl von $b = 100$ Samples aus insgesamt 49800 Samples eine Laufzeit von mehreren Stunden auf. In dieser Zeit hätte bereits ein großes Vielfaches der 100 Samples gelabelt werden können, weshalb der Algorithmus nicht weiter verfolgt wurde.

Um DeepAL effizienter zu machen, konzentriert sich ein weiterer Bereich der Forschung darauf, die Nutzung der unzureichenden Menge gelabelter Trainingsdaten zu verbessern, ohne die Kosten für das Labeling neuer Daten zu erhöhen. Das CEAL (Cost-Effective Active Learning), entwickelt von K. Wang et al. [WZL⁺17], fügt dem Trainingsdatensatz weitere Samples hinzu, denen ein sehr geringes Uncertainty Estimate vorhergesagt wird. Sie werden nicht durch ein Orakel gelabelt, stattdessen werden die Prädiktionen des Modells als Pseudo-Labels verwendet, da diese sehr wahrscheinlich korrekt sind. J.-J. Zhu et al. [ZB17] und T. Tran et al. [TDRC19] verfolgen einen anderen Ansatz, bei dem Samples, die zum Trainingsdatensatz hinzugefügt werden sollen, generiert werden. Die erzeugten Samples sind informativer als es durch die Auswahl vorhandener Daten der Fall ist, da sie gezielt so konstruiert werden, dass sie eine sehr hohe prädiktive Unsicherheit haben. Die Erweiterung des Trainingsdatensatzes durch generierte Daten kann, zu

einem höheren Performanacewachstum führen als die Verbesserung der Auswahlstrategie. [RXC⁺21]

Zusammenfassend ist festzustellen, dass der verringerte Labelaufwand beim DeepAL oft mit einem erhöhten Rechenaufwand verbunden ist. Es stellt sich die Frage, ob die Kosten für den Nutzen bei Volkswagen akzeptabel sind. Um dies einzuschätzen, wäre es hilfreich, wenn mehrere Datensätze Informationen über die Dauer des Labelings durch den Menschen bereitstellen könnten [GK20]. Die Effizienz von Optimierungsalgorithmen und Uncertainty Estimation Methoden ist besonders wichtig, um den Einsatz des DeepAL bei Volkswagen zu ermöglichen. Auch die optimale Gewichtung von Unsicherheit und Diversität bei hybriden Auswahlstrategien ist schwer allgemein festzustellen und müsste speziell für die Anwendungen bei Volkswagen untersucht werden. Mit der zufälligen Wahl eines Samples von der Pareto-Front in Kapitel 4.3.2 kann diese Entscheidung zunächst umgangen werden. Wird lediglich die Unsicherheit als Auswahlkriterium verwendet, zeigt sich erst nach mehreren Iteration eine Verbesserung beim Netzwerktraining, was nur nach langer Zeit für die Netzwerke im Smart.Production:Lab nützlich sein kann. Die Erweiterung des Trainingsdatensatzes mit generierten Samples könnte hohes Potenzial bei Volkswagen haben. Durch die Generierung eines Batches können die Kosten für die Suche eines geeigneten Batches aus einer großen Datenmenge vermieden werden, was Raum für zukünftige Untersuchungen schafft.

Im Rahmen dieser Bachelorarbeit konnte noch nicht herausgefunden werden, wie das aktive Lernen am besten bei Volkswagen praktiziert werden kann. Allerdings wurden die Ziele dieser Arbeit erreicht, indem die Uncertainty Estimation Methoden verglichen und deren Verwendung für das unsicherheitsbasierte DeepAL erfolgreich untersucht wurde. Zukünftige Arbeiten des Smart.Production:Lab können sich auf die Suche nach effizienten Techniken für das aktive Lernen fokussieren. Mit dem NUC ist eine schnelle Methode zur Bewertung der Unsicherheit ausgewählter Samples für die Klassifizierung gefunden worden, die nun von verschiedenen Techniken und Auswahlstrategien des aktiven Lernens genutzt werden kann. Die Untersuchung von Methoden zur Uncertainty Estimation ist allerdings noch für weitere Problemstellungen neben der Klassifizierung interessant.

Anhang

A. Tabellen

	CNN c10 ₁₀₀	CNN c10 ₁₀₀₀	CNN c10 ₁₀₀₀₀	CNN c10	CNN c100	Eff Cars
SE	0.000	0.000	0.000	0.000	0.000	0.000
MCD PE	0.004	0.002	0.001	0.001	0.002	0.001
MCD MI	0.002	0.002	0.001	0.001	0.002	0.001
Bag PE	0.009	0.003	0.002	0.002	0.005	0.001
Bag MI	0.024	0.004	0.009	0.003	0.006	0.002
DA PE	0.003	0.002	0.002	0.002	0.005	0.002
DA MI	0.017	0.020	0.010	0.004	0.005	0.002
NUC Tr	0.035	0.015	0.006	0.003	0.002	0.000
NUC Va	-	0.023	0.005	0.000	0.000	0.002

(a) Standardabweichungen AUOCs

	CNN c10 ₁₀₀	CNN c10 ₁₀₀₀	CNN c10 ₁₀₀₀₀	CNN c10	CNN c100	Eff Cars
SE	0.000	0.000	0.000	0.000	0.000	0.000
MCD PE	0.004	0.005	0.004	0.005	0.003	0.003
MCD MI	0.003	0.005	0.004	0.003	0.003	0.004
Bag PE	0.008	0.003	0.003	0.005	0.005	0.004
Bag MI	0.011	0.003	0.009	0.006	0.006	0.008
DA PE	0.005	0.003	0.002	0.006	0.009	0.010
DA MI	0.010	0.018	0.012	0.004	0.006	0.013
NUC Tr	0.030	0.018	0.008	0.001	0.002	0.000
NUC Va	-	0.023	0.005	0.001	0.000	0.009

(b) Standardabweichungen AUPRs

Tabelle 1: Standardabweichungen für die AUOCs und AUPRs aller Methoden zur Uncertainty Estimation (Abkürzungen: SE - Softmaxentropie; PE - prädiktive Entropie; MI - Mutual Information; MCD - MC Dropout; Bag - Bagging; DA - Data Augmentation; NUC Tr - NUC trainiert auf Trainingsdaten; NUC Va - NUC trainiert auf Validierungsdaten; c10 - Cifar10; c100 - Cifar100; c10_x - x Trainingsdaten von Cifar10; Eff - EfficientNet-B3)

B. Calibration Plots

Im Folgenden sind Calibration Plots von Certainty Estimates für jedes Modell dargestellt. Betrachtet werden der MC Dropout Ansatz, das Bagging und Data Augmentation Ensemble, die NUC-Methode (trainiert auf Prädiktionen für Trainingsdaten bzw. Validierungsdaten) und die Softmaxentropie. Beim MC Dropout und den Ensembles wird zwischen prädiktiver Entropie (PE) und Mutual Information (MI) unterschieden.

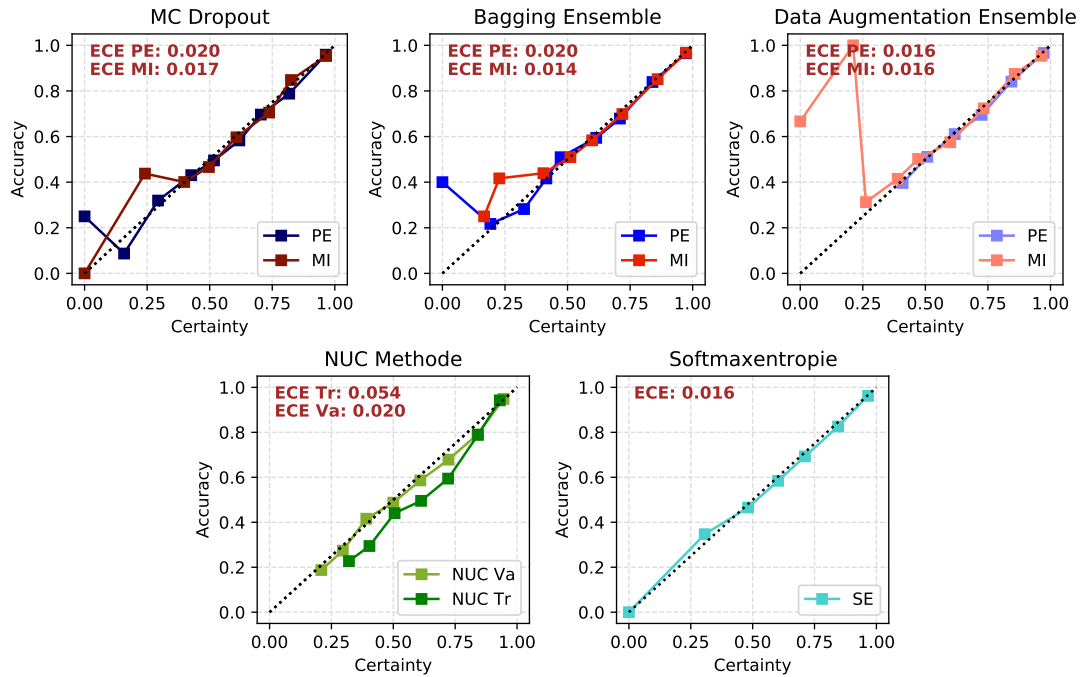


Abbildung 2: CNN für Cifar10.

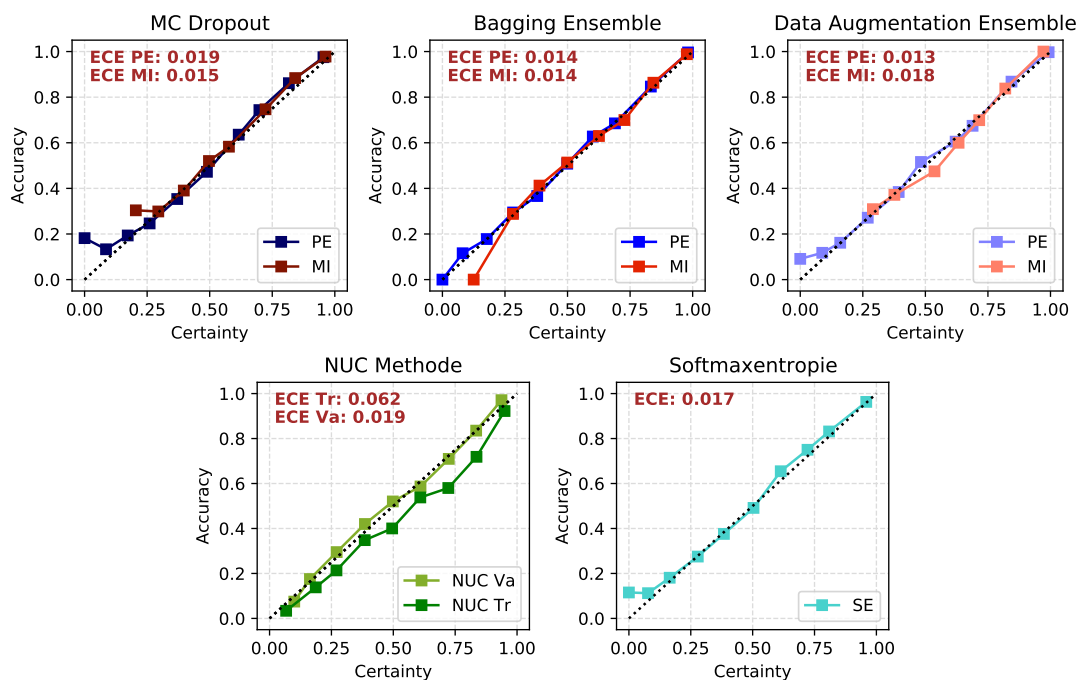


Abbildung 3: CNN für Cifar100.

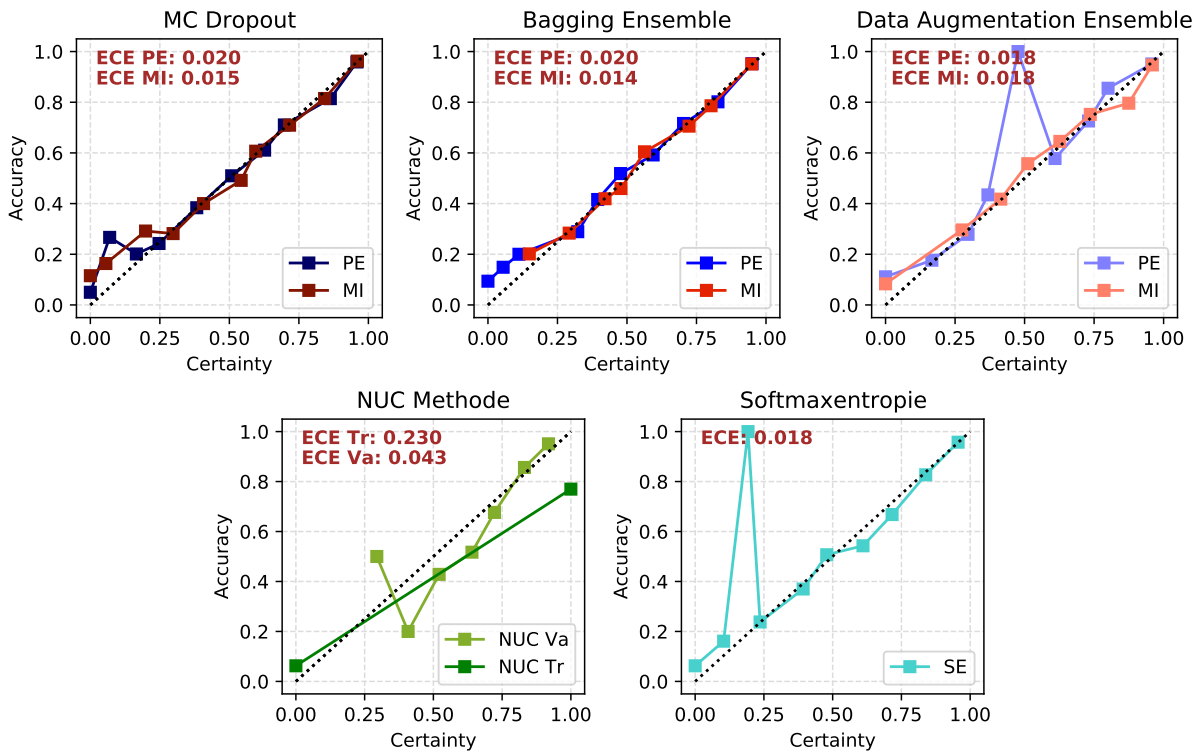


Abbildung 4: EfficientNet-B3.

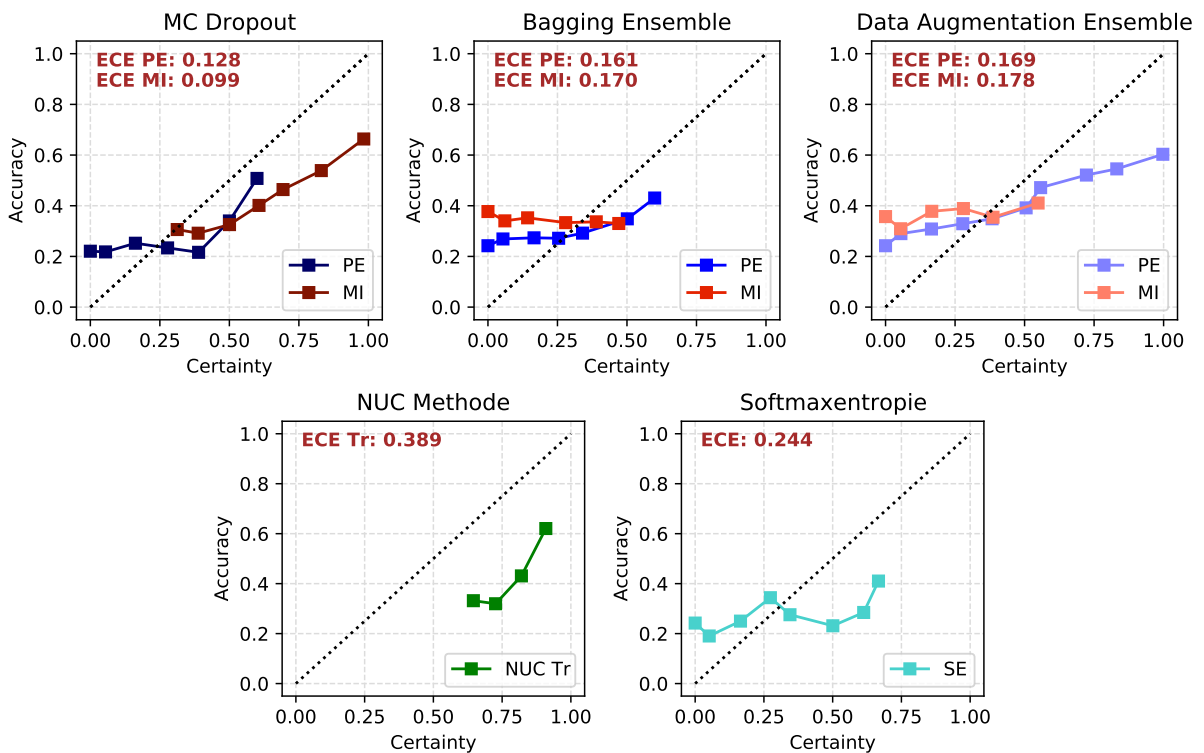


Abbildung 5: CNN trainiert auf 100 Bildern von Cifar10.

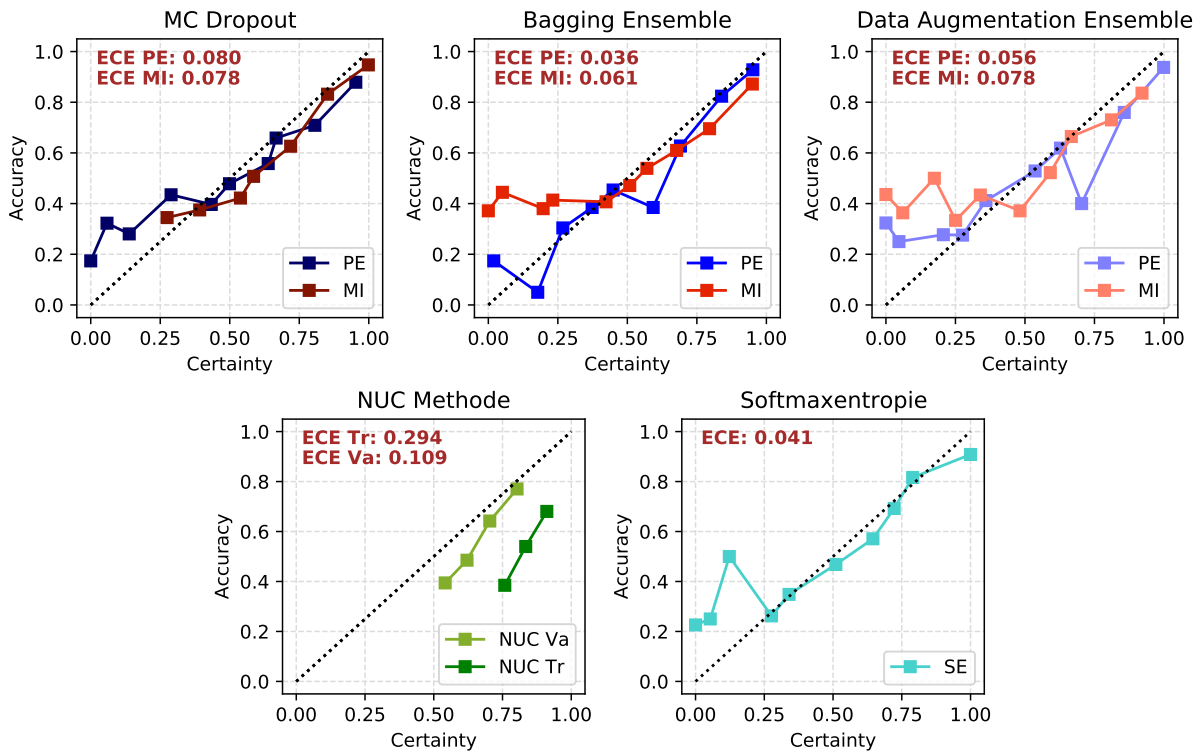


Abbildung 6: CNN trainiert auf 1000 Bildern von Cifar10.

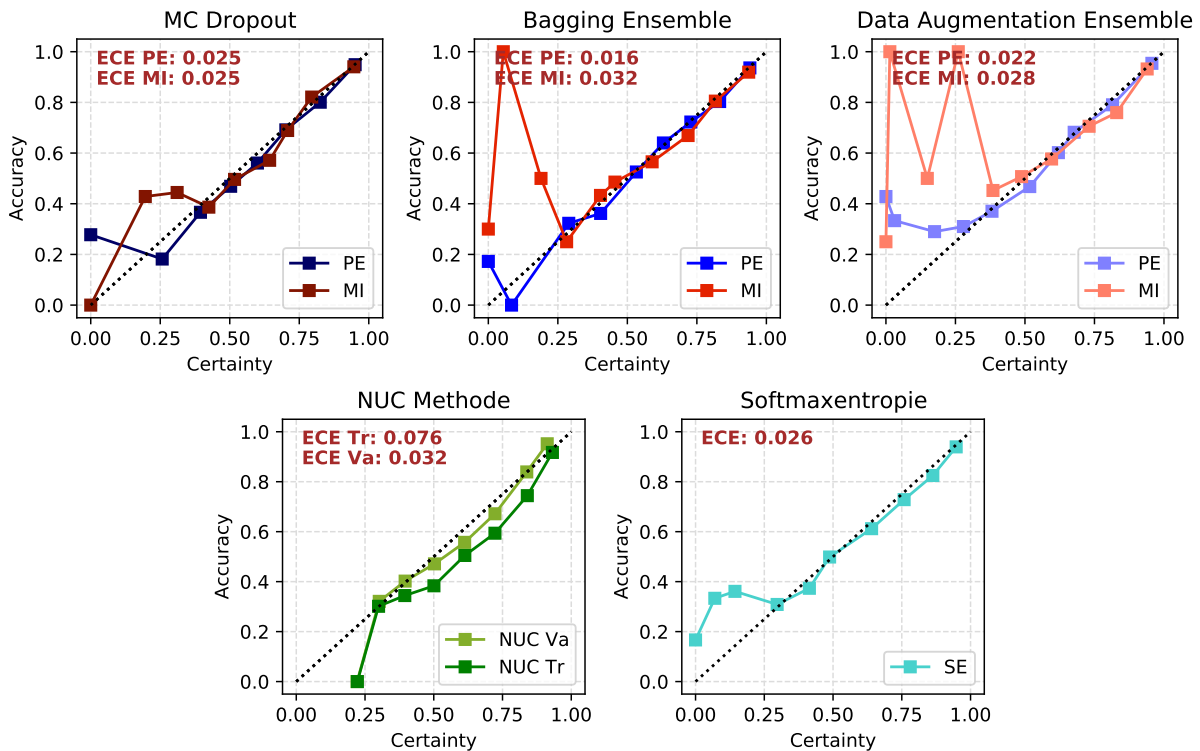


Abbildung 7: CNN trainiert auf 10000 Bildern von Cifar10.

C. ROC- und PR-Kurven

Dargestellt sind die ROC- und PR-Kurven für Uncertainty Estimates der Testdaten von Modellen, die nicht bereits in Kapitel 3.3.2 betrachtet wurden.

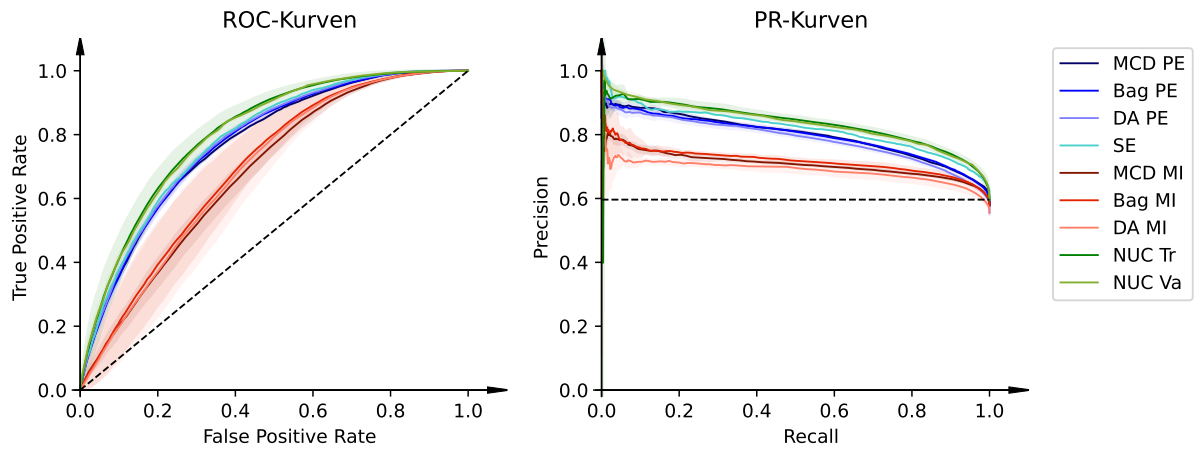


Abbildung 8: CNN für Cifar100.

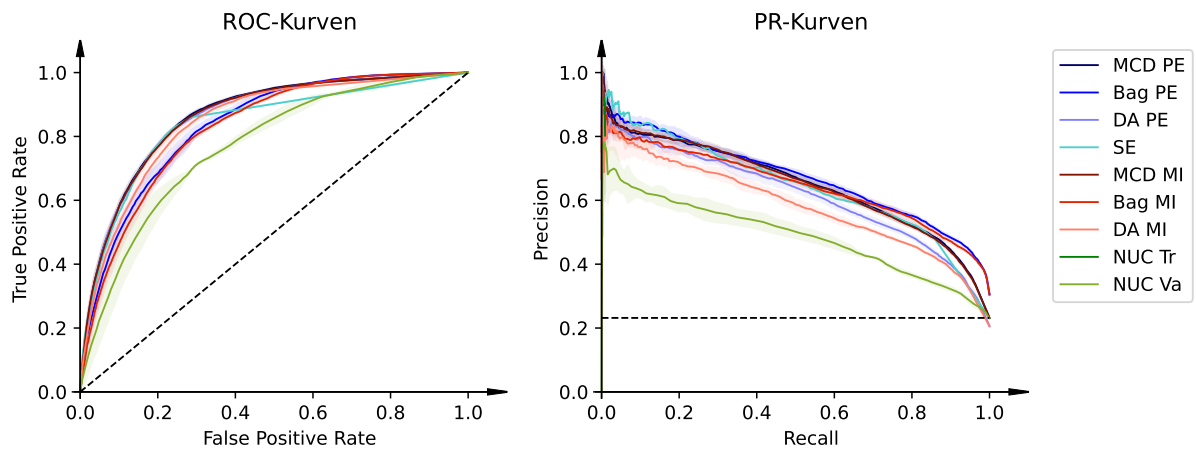


Abbildung 9: EfficientNet-B3.

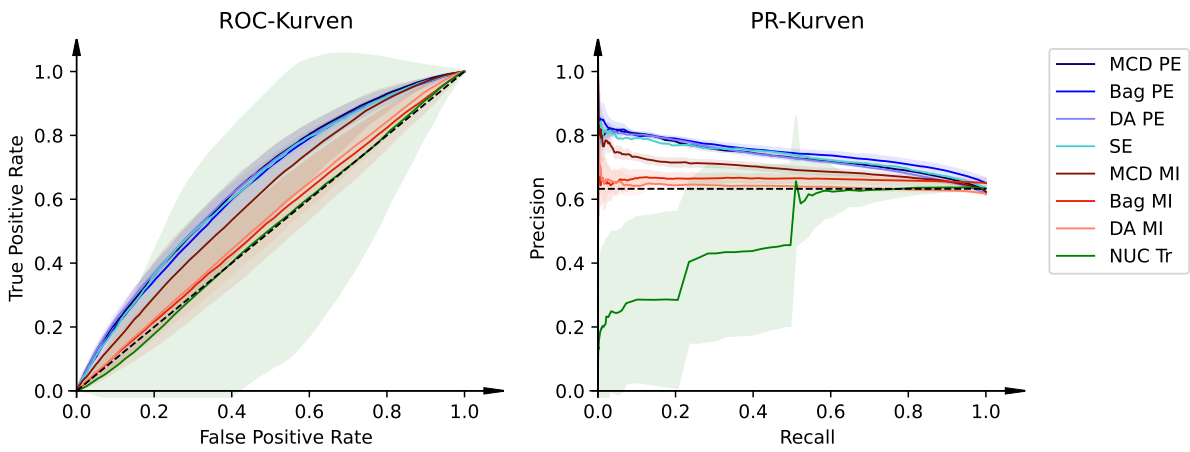


Abbildung 10: CNN trainiert auf 100 Bildern von Cifar10.

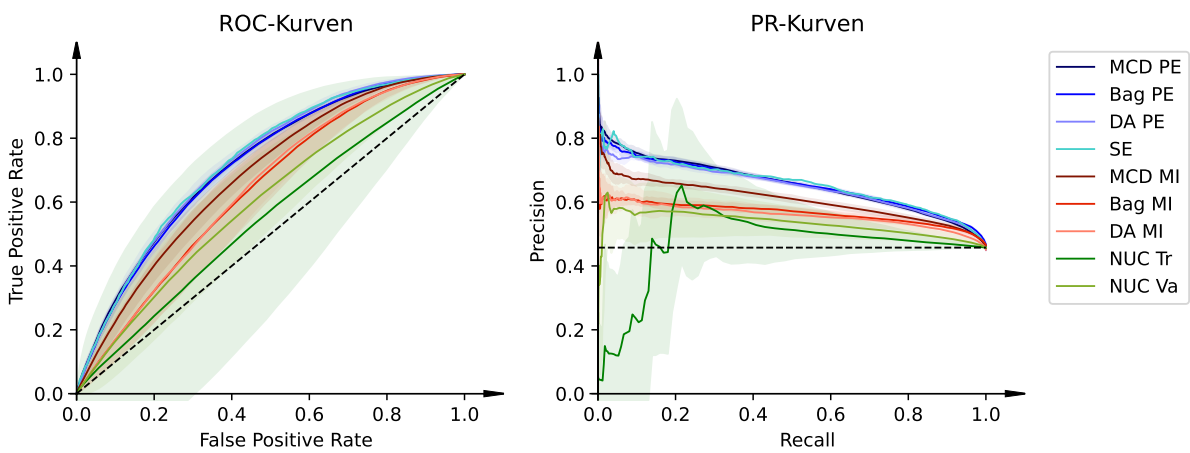


Abbildung 11: CNN trainiert auf 1000 Bildern von Cifar10.

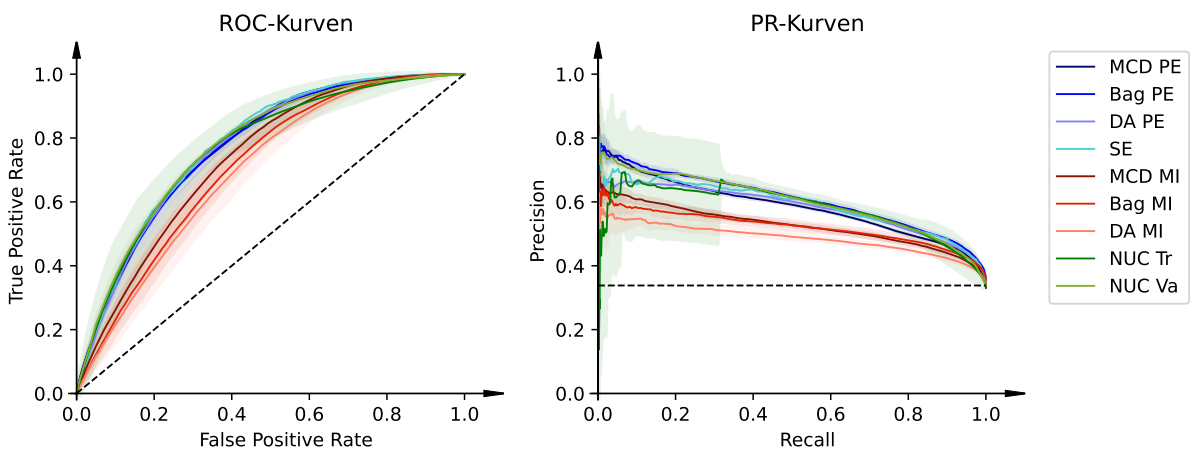


Abbildung 12: CNN trainiert auf 10000 Bildern von Cifar10.

D. Precision, Specificity und Recall der kalibrierten Uncertainty Estimates

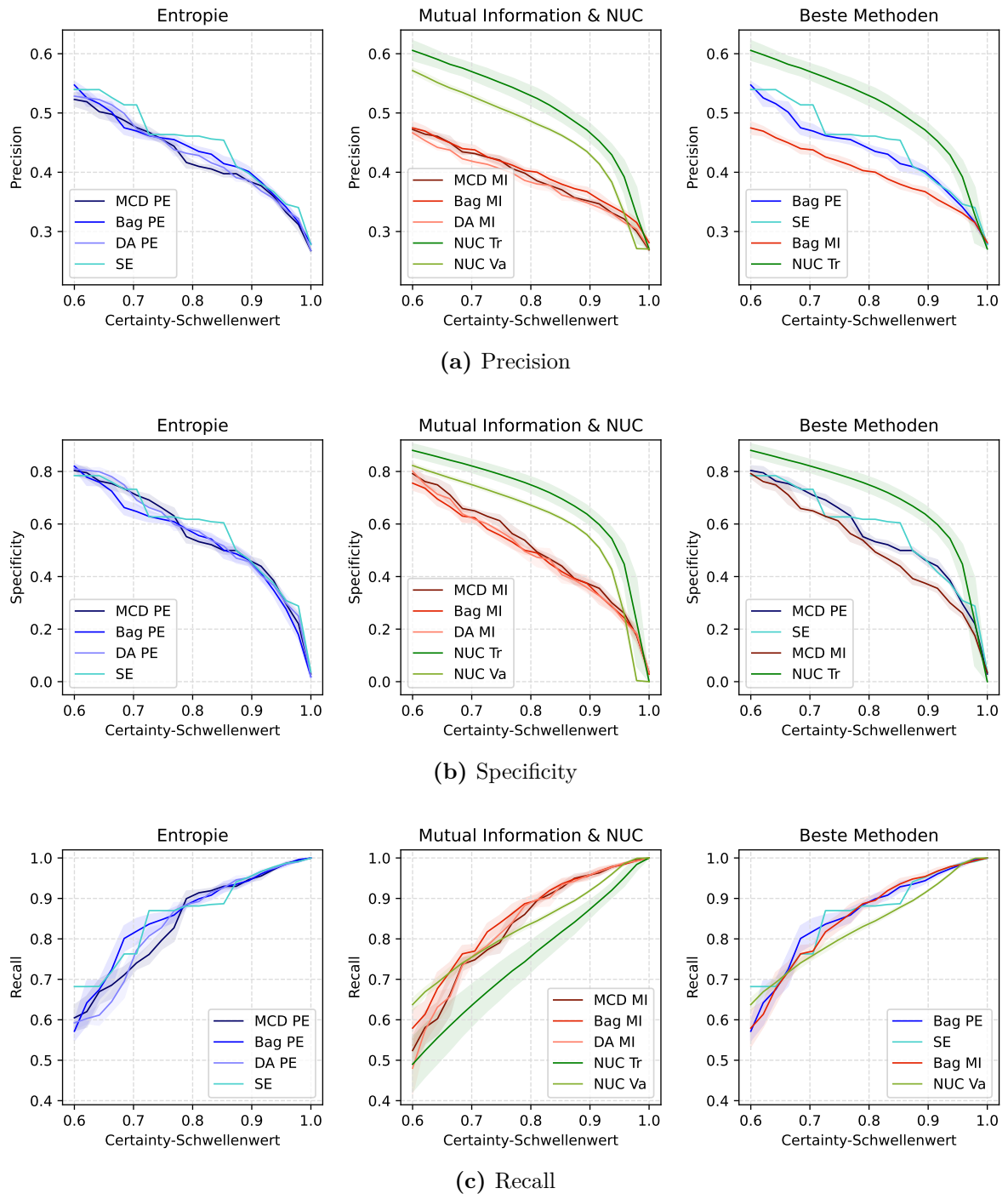


Abbildung 13: Darstellung der Metriken Precision, Specificity und Recall für die Uncertainty Estimation Methoden als Funktion von Certainty-Schwellenwerten für das CNN c10. Funktionswerte sind die Mittelwerte der Metriken nach fünffacher Ausführung der Methoden. Standardabweichungen sind leicht schattiert visualisiert.

Literaturverzeichnis

- [ABA⁺21] J. Antorán, U. Bhatt, T. Adel, A. Weller, and J. M. Hernández-Lobato. Getting a CLUE: A Method for Explaining Uncertainty Estimates. *International Conference on Learning Representations*, 2021.
- [ALMV20] A. Ashukha, A. Lyzhov, D. Molchanov, and D. Vetrov. Pitfalls of In-Domain Uncertainty Estimation and Ensembling in Deep Learning. *International Conference on Learning Representations*, 2020.
- [ASA⁺22] H. Asgharnezhad, A. Shamsi, R. Alizadehsani, A. Khosravi, S. Nahavandi, Z. A. Sani, D. Srinivasan, and S. M. S. Islam. Objective evaluation of deep uncertainty predictions for COVID-19 detection. *Scientific Reports*, 2022.
- [COR⁺16] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. *arXiv:1604.01685v2 [cs.CV]*, 2016.
- [CVC23] G. Csurka, R. Volpi, and B. Chidlovskii. Semantic Image Segmentation: Two Decades of Research. *arXiv:2302.06378v1 [cs.CV]*, 2023.
- [CWo⁺21] S. Cygert, B. Wróblewski, R. Słowiński, K. Woźniak, and A. Czyżewski. Closer Look at the Uncertainty Estimation in Semantic Segmentation under Distributional Shift. *arXiv:2106.00076v2 [cs.CV]*, 2021.
- [DDS⁺09] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.
- [DG06] J. Davis and M. Goadrich. The Relationship Between Precision-Recall and ROC Curves. *International Conference on Machine Learning (ICML)*, 2006.
- [DG19] I. Düntsch and G. Gediga. Confusion matrices and rough set data analysis. *arXiv:1902.01487v1 [cs.LG]*, 2019.
- [Fen21] D. Feng. *Uncertainty Estimation for Object Detection Using Deep Learning Approaches*. PhD thesis, Universität Ulm, 2021.
- [FHL19] S. Fort, H. Hu, and B. Lakshminarayanan. Deep Ensembles: A Loss Landscape Perspective. *arXiv:1912.02757v2 [stat.ML]*, 2019.
- [FK15] P. A. Flach and M. Kull. Precision-Recall-Gain Curves: PR Analysis Done Right. *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, 2015.
- [Gal16] Y. Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

- [GBC17] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. The MIT Press, Massachusetts Institute of Technology, 2017.
- [GEY21] I. Galil and R. El-Yaniv. Disrupting Deep Uncertainty Estimation Without Harming Accuracy. *35th Conference on Neural Information Processing Systems*, 2021.
- [GG16] Y. Gal and Z. Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *International Conference on Machine Learning*, pages 2–4, 2016.
- [GiNCF17] M. Gorriz, X. Giro i Nieto, A. Carlier, and E. Faure. Cost-Effective Active Learning for Melanoma Segmentation. *arXiv:1711.09168v2 [cs.CV]*, 2017.
- [GK20] S. A. Golestaneh and K. M. Kitani. Importance of Self-Consistency in Active Learning for Semantic Segmentation. *arXiv:2008.01860v1 [cs.CV]*, 2020.
- [GPSW17] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On Calibration of Modern Neural Networks. *arXiv:1706.04599v2 [cs.LG]*, 2017.
- [GTA⁺22] J. Gawlikowski, C. R. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher, M. Shahzad, W. Yang, R. Bamler, and X. X. Zhu. A Survey of Uncertainty in Deep Neural Networks. *arXiv:2107.03342v3 [cs.LG]*, 2022.
- [GW19] P. Ganti and S. L. Waslander. Network Uncertainty Informed Semantic Feature Selection for Visual SLAM. *arXiv:1811.11946v2 [cs.CV]*, 2019.
- [HAB19] M. Hein, M. Andriushchenko, and J. Bitterwolf. Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [HG18] D. Hendrycks and K. Gimpel. A Baseline For Detecting Mmisclassified and Out-of-Distribution Examples in Neural Networks. *arXiv:1610.02136v3 [cs.NE]*, 2018.
- [HKH⁺22] M. Hasan, A. Khosravi, I. Hossain, A. Rahman, and S. Nahavandi. Controlled Dropout for Uncertainty Estimation. *arXiv:2205.03109v1 [cs.LG]*, 2022.
- [HRG⁺22] A. Hein, S. Röhrle, T. Grobel, M. Lengle, N. Hafez, M. Knopp, C. Klenk, D. Heim, O. Hayden, and K. Diepold. A Comparison of Uncertainty Quantification Methods for Active Learning in Image Classification. *2022 International Joint Conference on Neural Networks (IJCNN)*, 2022.
- [HS21] C. J. Holder and M. Shafique. Efficient Uncertainty Estimation in Semantic Segmentation via Distillation. *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2021.
- [HSRW20] M. Henne, A. Schwaiger, K. Roscher, and G. Weiss. Benchmarking Uncertainty Estimation Methods for Deep Learning With Safety-Related Metrics. *Workshop on Artificial Intelligence Safety, SafeAI 2020*, 2020.

- [KF14] M. Kull and P. A. Flach. Reliability maps: a tool to enhance probability estimates and improve classification accuracy. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2014.
- [Kri09] A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Master’s thesis, University of Toronto, 2009.
- [KSDF13] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3D Object Representations for Fine-Grained Categorization. *4th IEEE Workshop on 3D Representation and Recognition, at ICCV 2013*, 2013.
- [KvAG19] A. Kirsch, J. v. Amersfoort, and Y. Gal. Efficient and Diverse Batch Acquisition for Deep Bayesian Active Learning. *33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada*, 2019.
- [LPB17] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. *Advances in Neural Information Processing Systems (NeurIPS)*, page 6402–6413, 2017.
- [MvATG21] J. Mukhoti, J. v. Amersfoort, P. H. S. Torr, and Y. Gal. Deep Deterministic Uncertainty for Semantic Segmentation. *arXiv:2111.00079v1 [cs.CV]*, 2021.
- [NSFC21] A. Navon, A. Shamsian, E. Fetaya, and G. Chechik. Learning the Pareto Front with Hypernetworks. *arXiv:2010.04104v2 [cs.LG]*, 2021.
- [OFR⁺19] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek. Can You Trust Your Model’s Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift. *Advances in Neural Information Processing Systems*, 2019.
- [ON15] K. O’Shea and R. Nash. An Introduction to Convolutional Neural Networks. *arXiv:1511.08458v2 [cs.NE]*, 2015.
- [Pau03] J. A. Paulos. *A Mathematician Plays the Stock Market*. Basic Books, New York, 2003.
- [Pri15] L. Priese. *Computer Vision: Einführung in die Verarbeitung und Analyse digitaler Bilder*. Springer Vieweg, Koblenz, 2015.
- [RM19] T. Ramalho and M. Miranda. Density estimation in representation space to predict model uncertainty. *arXiv:1908.07235v2 [cs.LG]*, 2019.
- [RXC⁺21] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, B. B. Gupta, X. Chen, and X. Wang. A Survey of Deep Active Learning. *arXiv:2009.00236v2 [cs.LG]*, 2021.
- [Sch11] P. Schnäbele. Bayes-Statistik und konjugierte Verteilungen. Bachelorarbeit, Karlsruher Institut für Technologie (KIT), 2011.
- [SHK⁺14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15, pages 1929–1958, 2014.
- [Sie20] M. Sielvogel. Deep Active Learning for Classification Tasks. Master’s thesis, Technische Universität München, 2020.

- [TDRC19] T. Tran, T.-T. Do, I. D. Reid, and G. Carneiro. Bayesian Generative Active Deep Learning. *Proceedings of the 36th International Conference on Machine Learning*, 2019.
- [TL20] M. Tan and Q. V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv:1905.11946v5 [cs.LG]*, 2020.
- [Wes23] P. S. Wespe. Integration der Metrik Uncertainty in die Volkswagen Vision Workbench. Bachelorarbeit, Ostfalia Hochschule für angewandte Wissenschaften, 2023.
- [WS14] D. Wang and Y. Shang. A New Active Labeling Method for Deep Learning. *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems*, 2014.
- [WZL⁺17] K. Wang, D. Zhang, Y. Li, R. Zhang, and L. Lin. Cost-Effective Active Learning for Deep Image Classification. *arXiv:1701.03551v1 [cs.CV]*, 2017.
- [YQC⁺17] C. Yin, B. Qian, S. Cao, X. Li, J. Wei, Q. Zheng, and I. Davidson. Deep Similarity-Based Batch Mode Active Learning with Exploration-Exploitation. *2017 IEEE International Conference on Data Mining (ICDM)*, 2017.
- [ZB17] J.-J. Zhu and J. Bento. Generative Adversarial Active Learning. *arXiv:1702.07956v5 [cs.LG]*, 2017.
- [ZQD⁺20] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. A Comprehensive Survey on Transfer Learning. *arXiv:1911.02685v3 [cs.LG]*, 2020.
- [ZYZ⁺22] H. Zeng, Z. Yue, Y. Zhang, Z. Kou, L. Shang, and D. Wang. On Attacking Out-Domain Uncertainty Estimation in Deep Neural Networks. *31st International Joint Conference on Artificial Intelligence (IJCAI-22)*, 2022.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Bachelorarbeit selbstständig und ausschließlich unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Stellen sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form weder einer anderen Prüfungsbehörde vorgelegt oder noch anderweitig veröffentlicht.

A handwritten signature in blue ink, appearing to read 'H. Lichtenberg', with a long horizontal stroke extending to the right.

Magdeburg, den 05. April 2023