**Technische Universität Ilmenau**
Fakultät für Informatik und Automatisierung
Fachgebiet Neuroinformatik und Kognitive Robotik

# Schätzung der Unsicherheiten in Deep-Learning-basierten Lernverfahren

Masterarbeit zur Erlangung des akademischen Grades Master of Science

**Benedict Stephan**

Betreuer: Dr.-Ing. Volker Stephan, STEAG Energy Services GmbH
Dipl.-Inf. Ronny Stricker, FG Neuroinformatik und Kognitive Robotik
M.Sc. Daniel Seichter, FG Neuroinformatik und Kognitive Robotik

Verantwortlicher Hochschullehrer:
Prof. Dr. H.-M. Groß, FG Neuroinformatik und Kognitive Robotik

Die Masterarbeit wurde am 06.03.2019 bei der Fakultät für Informatik und Automatisierung der Technischen Universität Ilmenau eingereicht.

Ich bedanke mich bei der Firma STEAG Energy Services GmbH, welche mich während dieser Arbeit finanziell unterstützt hat, was es mir ermöglichte mich ganz auf diese Arbeit zu konzentrieren. Besonderer Dank gebührt dabei Volker Stephan, durch den diese Zusammenarbeit erst ermöglicht wurde. Weiterhin bedanke ich mich bei Ronny Stricker und Daniel Seichter, die mich trotz der theoretischen Natur dieser Arbeit stets mit Ratschlägen unterstützt und bei der Arbeit geleitet haben. Bei Dustin Aganian und Karl Amende bedanke ich mich, da sie auch die frühen Versionen der schriftlichen Ausarbeitung dieser Arbeit zu lesen bekamen, um mir bei der Ausformulierung helfen.

Erklärung: „Hiermit versichere ich, dass ich diese wissenschaftliche Arbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe. Alle von mir aus anderen Veröffentlichungen übernommenen Passagen sind als solche gekennzeichnet.“

Ilmenau, 06.03.2019 .....

Benedict Stephan

# Inhaltsverzeichnis

**1 Einleitung** **1**

**2 Theoretische Grundlagen** **5**

2.1 Typen von Unsicherheiten . . . . . 5

2.2 Dropout . . . . . 7

2.3 Receiver Operating Characteristic (ROC) . . . . . 8

2.4 f-Divergenzen . . . . . 10

2.4.1 Kullback-Leibler Divergenz . . . . . 11

2.4.2 $\alpha$-Divergenz . . . . . 14

**3 Schätzung von Unsicherheiten** **17**

3.1 Bayes'scher Ansatz . . . . . 18

3.2 Approximierende bayessche Ansätze . . . . . 19

3.2.1 Optimierungs-basiert . . . . . 19

3.2.2 Sampling-basiert . . . . . 23

3.3 Nicht-bayessche Ansätze . . . . . 25

3.4 Bewertung von Unsicherheiten . . . . . 32

3.4.1 Kalibrierungskurve . . . . . 32

3.4.2 Log-Likelihood . . . . . 40

3.5 Zusammenfassung . . . . . 43

**4 Verwendete Datensätze** **45**

4.1 Synthetische Daten . . . . . 45

4.2 Prozessdaten der STEAG GmbH . . . . . 50

4.2.1 Vorverarbeitung . . . . . 52

4.2.2 Vorgehen bei der Anomaliedetektion . . . . . 53

**5 Experimentelle Untersuchungen 59**
5.1 Synthetische Daten . . . 60
5.1.1 Modellarchitektur . . . 61
5.1.2 Konvergenz der Modelle und deren Ausgaben . . . 66
5.1.3 Modellunsicherheit . . . 72
5.1.4 Datenunsicherheit . . . 85
5.2 STEAG Prozessdaten . . . 91
5.2.1 Modellarchitektur . . . 92
5.2.2 Allgemeine Untersuchung der Unsicherheiten . . . 100
5.2.3 Anomaliedetektion . . . 104
5.3 Zusammenfassung . . . 110
5.3.1 Zusammenfassung - Synthetische Daten . . . 110
5.3.2 Zusammenfassung - STEAG . . . 111

**6 Zusammenfassung und Ausblick 113**
6.1 Zusammenfassung . . . 113
6.2 Ausblick . . . 114

**A Ergänzende Unterlagen 117**
A.1 Synthetische Daten . . . 117
A.1.1 Lernkurven der Modelle . . . 117
A.1.2 Auswirkungen der Aktivierungsfunktion auf Modellunsicherheiten . . . 118
A.1.3 Modellunsicherheit in Abhängigkeit der Sample-Anzahl . . . 120
A.1.4 Concrete-Dropout . . . 121
A.1.5 Auswirkungen der Minibatch-Größe . . . 123
A.1.6 Datensatztests für kleinere Dropout-Raten . . . 123
A.2 STEAG . . . 125
A.2.1 STEAG - Modellvergleich . . . 125
A.3 Herleitungen . . . 132

**Literaturverzeichnis 141**

# Abkürzungsverzeichnis

| | |
|---|---|
| **SES** | STEAG Energy Services GmbH |
| **KL** | Kullback-Leibler-Divergenz |
| **VI** | Variational Inference |
| **EP** | Expectation Propagation |
| **MDN** | Mixture Density Networks |
| **MCMC** | Marcov Chain Monte Carlo |
| **NUTS** | No-U-Turn Sampler |
| **SPC** | stochastic process control |
| **SPRT** | sequential probability ratio test |
| **MSE** | Mean Squared Error |
| **ROC** | Receiver Operating Characteristic |
| **FPR** | false positive rate |
| **TPR** | true positive rate |
| **AUC** | area under curve |
| **ECE** | Expected Calibration Error |
| **KPI** | key performace indicator |

# Notation

| | |
|---|---|
| $\underline{\mathbf{x}}$ | Eingabevektor: $\underline{\mathbf{x}}^T = (x_1, \ldots, x_j, \ldots, x_n)$ |
| $\underline{\mathbf{x}}^*$ | Unbekannter Eingabevektor: $\underline{\mathbf{x}}^{*T} = (x_1^*, \ldots, x_j^*, \ldots, x_n^*)$ |
| $\underline{\mathbf{y}}$ | Ausgabevektor: $\underline{\mathbf{y}}^T = (y_1, \ldots, y_j, \ldots, y_n)$ |
| $\underline{\mathbf{t}}$ | Teachervektor: $\underline{\mathbf{t}}^T = (t_1, \ldots, t_j, \ldots, t_n)$ |
| $D_f(p \,\|\| \, q)$ | f-Divergenz zwischen Verteilungen $p$ und $q$ |
| $D_{KL}(p \,\|\| \, q)$ | Kullback-Leibler-Divergenz zwischen $p$ und $q$ |
| $L(t \mid x,\theta)$ | Likelihood des Teacherwertes $t$ unter dem Input $x$ und den Model-Parameters $\theta$ |
| $\mathbf{D}$ | Datensatz: $\mathbf{D} = \{(\underline{\mathbf{t}}_i, \underline{\mathbf{x}}_i)\}_{i=1}^M$ |

# Kapitel 1

# Einleitung

Deep-Learning-basierte Lösungsansätze zeigen sich in einer Vielzahl an Problemfelder als sehr erfolgreich und übertreffen teilweise sogar den Menschen. Dabei finden sie auch Einsatz in Aufgabenbereichen, für die es gilt eine optimale Entscheidung zu treffen. Zum Beispiel werden sie eingesetzt, um Objekte in einem Bild zu detektieren, die Regelung eines komplexen Systems zu übernehmen oder selbständig ein Kraftfahrzeug zu steuern. In einigen dieser Aufgabenbereiche, wie beispielsweise dem autonomen Fahren, spielen Aspekte der Sicherheit eine große Rolle. Für diese Problemstellungen ist es von großem Vorteil eine Aussage über die Unsicherheit, welche bei einer gewissen Entscheidung besteht, treffen zu können. So sollte sich ein System beispielsweise nicht auf die Entscheidung eines Modells verlassen, wenn diese mit einer zu großen Unsicherheit behaftet ist, um ein fehlerhaftes oder sogar gefährliches Verhalten zu verhindern. Ein Mensch beispielsweise weiß in der Regel, wenn er sich bei einer Entscheidung sicher ist, oder ab wann er zugeben muss, dass eine Wahl praktisch zufällig getroffen wurde, da eine sichere Entscheidung auf Basis seines Wissensstandes nicht möglich war. Diese Unsicherheit beeinflusst den Entscheidungsprozess, da sich auf unsichere Aussagen nicht verlassen werden kann. Falls also ein autonomes System die Möglichkeit besitzen würde mitzuteilen, falls keine sichere Aussage möglich ist, dann können entsprechende Maßnahmen eingeleitet werden, um das Fehlen dieser Aussage zu kompensieren.
Allerdings ist die Anwendung von Unsicherheiten nicht nur auf diese Fälle beschränkt. Sie sind vielmehr vielseitig einsetzbar. Beispielsweise spielen sie eine Rolle in den Verfahren zur Hyperparametersuche für das Training eines Modells (bayessche Hyper-

parametersuche[1]). Auch im Bereich des Reinforcement Learning bieten Unsicherheiten einen Vorteil. Dort werden sie beispielsweise verwendet, um eine Aktion während des Trainings zu wählen, da Aktionen, bei welchen sich ein Aktor unsicher ist, weiter erprobt werden sollten, um einen Aktor zu trainieren, welcher sich in seinen Aktionen möglichst sicher ist.

Es ist also für eine Vielzahl an Problemstellungen vorteilhaft die Unsicherheiten, welche unweigerlich Teil eines Modells sind, zu extrahieren und zu berücksichtigen. Der Aspekt der Unsicherheit wurde allerdings im Bereich des Deep Learning weitestgehend vernachlässigt. Hier wird sich meist auf das trainierte Modell als Punktschätzung einer deterministischen Lösung verlassen, da damit dennoch ein großer Erfolg für verschiedenste Problemstellungen erreicht werden kann. Können also die Vorteile, die das Wissen um die Unsicherheiten mit sich bringen, mit der Leistungsfähigkeit tiefer neuronaler Netze verbunden werden, so können beispielsweise sicherere und robustere Systeme entwickelt werden. Deshalb befassen sich derzeit viele Forschungen auf das Gebiet der Unsicherheiten im Kontext des Deep Learning. So gibt es beispielsweise viele Bestrebungen Ansätze der bayesschen Methodiken, welche als Grundstein für die Arbeit mit Unsicherheiten stehen, in den Bereich des Deep Learning zu integrieren.

Ziel dieser Arbeit ist es nun die theoretischen Zusammenhänge und Grundlagen der Verfahren zur Schätzung von Unsicherheiten in Bezug auf Neuronale Netze aufzuarbeiten, um eine Grundlage für weitere Arbeiten und ein Verständnis für die Thematik der Unsicherheiten zu schaffen. Damit eine verlässliche Grundlage für weitere Arbeiten geschaffen werden kann, ist es wichtig die verwendeten Verfahren auf einem Problem zu erproben, für das möglichst viele Informationen vorliegen, damit die resultierenden Unsicherheiten möglichst gut bewertet und analysiert werden können. Deshalb besteht ein Großteil dieser Arbeit aus Untersuchungen auf einem selbst generierten Datensatz. Um dennoch praxisnahe Untersuchungen durchführen zu können, wird die Anwendung von Unsicherheiten in Zusammenarbeit mit der STEAG Energy Services GmbH (SES) für Messdaten eines industriellen Prozesses erprobt.

Im folgenden Kapitel werden allerdings zuerst wichtige theoretische Grundlagen erläutert, welche eine Basis für die Arbeit mit Unsicherheiten darstellt. Zudem werden Techniken und Werkzeuge aus dem Bereich des Deep Learning, welche in dieser Arbeit Verwendung finden, erläutert. Danach werden die verschiedenen Ansätze zur

[1] Beispiel einer aktuellen Verwendung dieser Verfahren ist AlphaGo[Chen et al., 2018]

Schätzung von Unsicherheiten in Kapitel 3 aufgearbeitet und ein Überblick über den State of the Art geliefert. In Kapitel 4 werden nachfolgend die in dieser Arbeit verwendeten Daten beschrieben. Dabei wird auf die Details der Erstellung beziehungsweise Verarbeitung der Datensätze eingegangen. Anschließend werden in Kapitel 5 die Experimente auf den einzelnen Datensätzen präsentiert und deren Ergebnisse interpretiert. Abschließend wird in Kapitel 6 eine Zusammenfassung dieser Arbeit und ein Ausblick für weiterführende Arbeiten geliefert.

# Kapitel 2

# Theoretische Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen, die als Voraussetzung für den Rest dieser Arbeit gelten, genannt und erläutert. Eine der wichtigsten Grundlagen stellt dabei die Definition der verwendeten Unsicherheiten dar. Neben diesen Definitionen werden zudem für diese Arbeit wichtige Verfahren für das Training und die Bewertung von Neuronalen Netzen erläutert. Zudem werden theoretische Begriffe genannt und definiert, welche eine Grundlage für Verfahren des State of the Art bilden.

## 2.1 Typen von Unsicherheiten

Um mit Unsicherheiten arbeiten zu können muss der Begriff „Unsicherheit“ zuerst definiert werden.
Unsicherheiten werden meist in Form einer Verteilung beschrieben. Die Ursache von Unsicherheit kann dabei auf viele verschiedene Gründe zurückgeführt werden. Laut [Kiureghian und Ditlevsen, 2009] gibt es dabei zwei grundlegende Gruppen, welche die verschiedenen Unsicherheiten ausreichend beschreiben.
In der Gruppe der *epistemischen* Unsicherheiten befinden sich all jene, welche sich durch das Verwenden von zusätzlichen Datenpunkten verringern lassen. Dazu gehört die Unsicherheit in der Schätzung von Modellparametern, wie zum Beispiel den Gewichten eines Neuronalen Netzes, und die Unsicherheit, die entsteht wenn ein Datenpunkt aus einer unbekannten Region des Inputraums stammt. Da sich diese Unsicherheiten meist auf das Model beziehen wird diese Art der Unsicherheit auch *Modellunsicherheit*[1]

[1] Eine weitere Bezeichnung ist *Unsicherheit* (engl. *uncertainty*) oder *reducible uncertainty*.

**Abbildung 2.1:** Visualisierung der zwei Haupttypen von Unsicherheiten.
*Zu sehen ist eine Ausgabe eines hypothetischen Modells in rot. Die schwarzen Punkte stellen die Trainingspunkte für das Model dar. Der Anstieg der Unsicherheit am linken Rand wird durch das vorhandene Rauschen in den Datenpunkten verursacht (Datenunsicherheit), während die Unsicherheit am linken Rand entsteht, da hier keine Trainingsbeispiele vorhanden sind und diese Region für das Model somit unbekannt ist (Modellunsicherheit).*

genannt.
Die Gruppe der *aleatorischen* Unsicherheiten umfasst diejenigen, welche sich durch zusätzliche Daten nicht verringern lassen. Beispielsweise Unsicherheiten in Zielwerten, welche durch einen Rauschterm entstehen. Auch mit mehr Daten kann ein Zielwert in diesem Fall nicht korrekt vorhergesagt werden, da dieser durch einen Zufallsprozess beeinflusst wird. Da diese Art der Unsicherheiten meist auf den verwendeten Daten basieren wird diese auch *Datenunsicherheit*[2] genannt. Dabei kann wiederum eine Unterscheidung zwischen *varianzhomogener* und *varianzheterogener* Datenunsicherheit gemacht werden. Diese unterscheiden sich dabei an der Art des Rauschens. Ist dieses für alle Datenpunkte identisch und somit unabhängig von der Eingabe, so handelt es sich um varianzhomogene Datenunsicherheit. Falls der Rauschterm abhängig von der Eingabe ist, so spricht man von varianzheterogener Datenunsicherheit. Zur Veranschaulichung ist in

[2] Eine weitere Bezeichnung ist *Risiko* (engl. *risk*) oder *irreducible uncertainty*

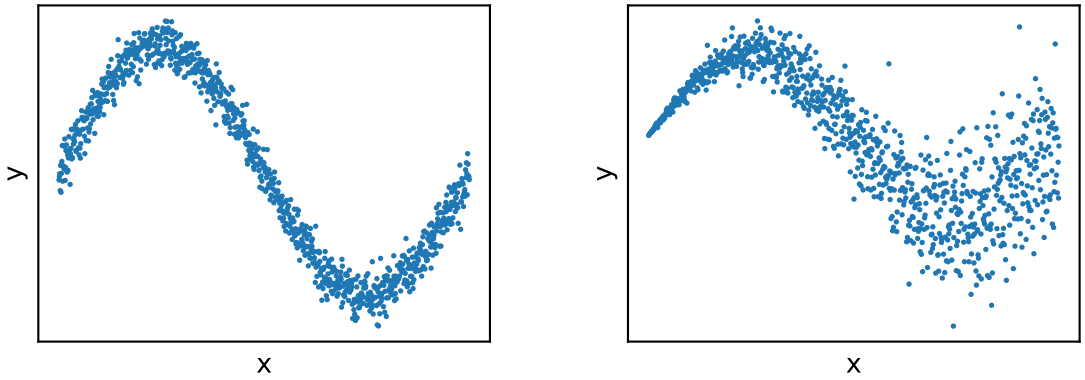



**Abbildung 2.2:** Visualisierung von varianzhomogener bzw. -heterogener Datenunsicherheit.
*Die beiden Abbildungen zeigen jeweils einen Ausschnitt der Sinus-Funktion. Auf der linken Seite wurde dieser durch eine normalverteilte Störgröße mit konstanter Standardabweichung verrauscht. Auf der rechten Seite wurde die Störgröße durch eine Normalverteilung mit Standardabweichung abhängig von der Eingabe generiert.*

Abbildung 2.2 ein Beispiel für ein varianzhomogenes bzw. -heterogenes Datenrauschen abgebildet.
Im Weiteren werden nur diese beiden Arten von Unsicherheit unterschieden und als Modellunsicherheit bzw. Datenunsicherheit bezeichnet.

## 2.2 Dropout

Bei Dropout handelt es sich um eine Technik im Deep-Learning-Bereich, die ursprünglich zur Regularisierung verwendet wird. Für diese Arbeit ist diese Technik besonders wichtig, da viele der in Abschnitt 3.2.1 (Seite 19) beschriebenen Verfahren auf Dropout basieren. Erstmals verwendet wurde diese Technik dabei in [Srivastava et al., 2014]. Grundsätzlicher Gedanke ist hierbei, dass in einem Lernschritt nicht das gesamte Netzwerk verwendet wird sondern ein Subnetz. Dieses wird erzeugt, indem Neuronen mit einer gewissen Wahrscheinlichkeit $p$ aus dem Netz entfernt werden. Die Abbildung 2.3 zeigt dies anhand einer schematischen Darstellung eines Neuronalen Netzes. In der praktischen Anwendung wird dabei die Ausgabe einer Schicht mit einer Dropoutmaske $\underline{\mathbf{M}}$ multipliziert, wobei $\underline{\mathbf{M}}$ eine Diagonalmatrix darstellt für deren Diagonale $\underline{\mathbf{v}}$ gilt

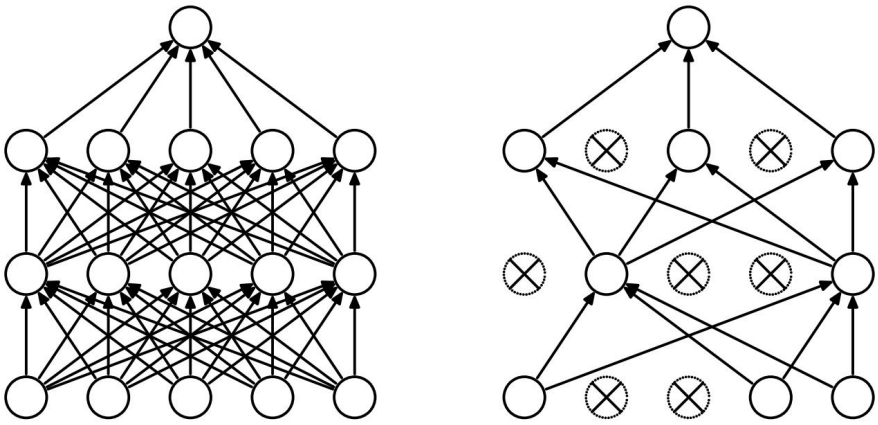


**Abbildung 2.3:** Darstellung des Prinzips hinter Dropout.
*Die Abbildung zeigt ein schematisches Beispiel eines Neuronales Netzes. Links: Standard Neuronales Netz. Rechts: Neuronales Netz nach der Anwendung einer Dropoutmaske. Durchkreuzte Neuronen sind dabei entfernt worden. Quelle:* [Srivastava et al., 2014]

$\underline{\mathbf{v}} \sim \text{Bern}(p)$. Die eigentliche Motivation dieses Verfahrens wurde dadurch gegeben, dass hier effektiv ein Ensemble von Subnetzen, welche sich ihre Gewichte teilen, trainiert wird. In der Anwendung wird damit über dieses Ensemble gemittelt. Zudem argumentieren die Autoren von [Srivastava et al., 2014], dass durch das zufällige Entfernen von Neuronen die Koadaption verhindert wird. Das heißt ein Neuron kann sich nicht auf die Ausgabe einer vorherigen Schicht verlassen und muss daher ein robusteres Verhalten erlernen.
Aufbauend auf diesem Prinzip wurden ähnliche Verfahren entwickelt, welche keine Neuronen entfernen, sondern einzelne Gewichte zwischen diesen [Wan et al., 2013], beides [Frazao und Alexandre, 2014] oder ganze Schichten [Huang et al., 2016].

## 2.3 Receiver Operating Characteristic (ROC)

Obwohl der Fokus dieser Arbeit auf der Problemstellung der Regression liegt wird die Receiver-Operating-Characteristic (ROC-Kurve) als Qualitätsmaß für binäre Klassifikationen verwendet. Sie bewertet anhand von Labeln, wie gut ein Modell eine Einteilung der Daten in die Positiv- bzw. Negativ-Klasse vornehmen kann. Dabei ist das Verhältnis der beiden Klassen unbedeutend.

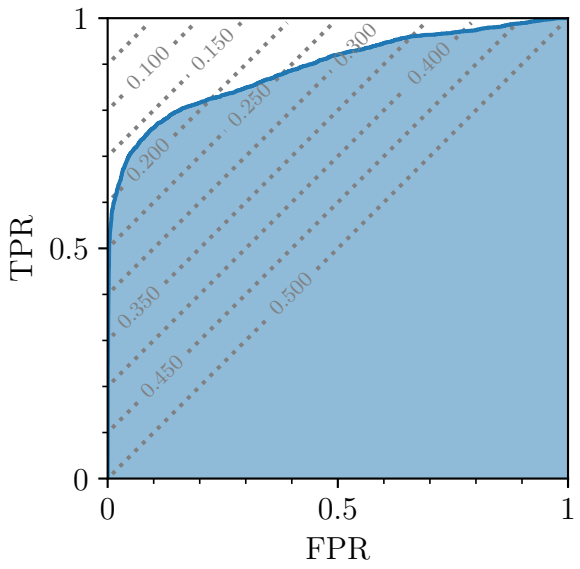



**Abbildung 2.4:** Aufbau eines ROC-Plots
*Zu sehen ist ein Beispiel einer ROC-Kurve. Dabei ist die Fläche, welche für den AUC-Score berechnet wird, eingefärbt.*

Im Zuge der Experimente zur Untersuchung der Modellunsicherheit in Abschnitt 5.1.3 (Seite 78) und der Anomaliedetektion in Abschnitt 5.2.3 (Seite 104) wird dieses Qualitätsmaß zur Bewertung der dort vorgenommen Klassifikation verwendet.
Für die Erstellung einer ROC-Kurve sind zum einen die Labeldaten $T = \{t_i\}_{i=1}^N$ und zum andern die Ausgabe des Klassifikators $Y = \{y_i\}_{i=0}^N$ notwendig. Die Ausgabe des Klassifikators $y_i$ hat dabei die Form eines Scores, das bedeutet sie beschreibt einen skalaren Wert. Je größer dieser Score, desto wahrscheinlicher ist es, dass ein Datenpunkt zu der Positiv-Klasse gehört. Für einen Schwellwert werden nun alle Punkte mit einem Score größer als dieser Schwellwert als Teil der Positiv-Klasse gezählt und alle Punkte mit einem Score unterhalb dieses Schwellwertes als Teil der Negativ-Klasse. Für die ROC-Kurve wird nun die Richtig-Positive-Rate (englisch: true positive rate (TPR)) gegen die Falsch-Positiv-Rate (englisch: false positive rate (FPR)) für verschiedene Schwellwerte abgetragen.
Abbildung 2.4 zeigt ein Beispiel einer ROC-Kurve eines hypothetischen Klassifikators. Dabei sind zusätzlich Höhenlinien für bestimmte Werte der Balanced Error Rate (BER)

abgetragen. Diese wird berechnet durch:

$$BER = \frac{1}{2}((1 - \text{TPR}) + \text{FPR}) \tag{2.1}$$

Für eine Bewertung, ohne sich für einen bestimmten Schwellwert entscheiden zu müssen, wird die Fläche unter der ROC-Kurve (englisch: area under curve (AUC)) bestimmt. Dieser Wert wird mit ROC AUC abgekürzt, allerdings wird in dieser Arbeit der Begriff AUC-Score verwendet. Der AUC-Score hat einen Wertebereich von [0,1], wobei ein AUC-Score von 0.5 das schlechteste Ergebnis darstellt, da dies einer zufälligen Verteilung der Scores $y_i$ entspricht.
Eine ROC-Kurve, welche sich durchgehend unter der Diagonale befindet, würde für alle Schwellwerte ein Ergebnis bedeutet, welches schlechter als das eines ratenden Systems ist. In diesem Falle kann allerdings die Definition der Positiv- bzw. Negativ-Klasse umgedreht werden. Dies sorgt für eine Spiegelung der ROC-Kurve an der Diagonalen, was in diesem Falle in einer ROC-Kurve resultiert, welche durchgehend über der Diagonalen verläuft. Das bedeutet, es kann davon ausgegangen werden, dass sich ROC-Kurven nie vollständig unter der Diagonale aufhalten.

## 2.4 f-Divergenzen

Eine Divergenz beschreibt ein Maß, welches zur Bestimmung des Unterschiedes zwischen zwei Verteilungen verwendet wird. In diesem Abschnitt werden die für spätere Verfahren wichtigen Divergenzen vorgestellt. Da diese Divergenzen verwendet werden, um eine Verteilung durch eine andere anzunähern, wird im Kontext dieses Abschnitts die zu approximierende Verteilung als $p(x)$, und die approximierende Verteilung als $q(x)$ bezeichnet. Diese Divergenzen bilden für die Verfahren, welche in Abschnitt 3.2.1 (Seite 19) beschrieben werden, die Grundlage für die Formulierung der dort verwendeten Optimierungsprobleme. Daher wird auf Besonderheiten bei der Verwendung zur Optimierung einer Verteilung eingegangen.

### 2.4.1 Kullback-Leibler Divergenz

Das am meisten verwendete Divergenzmaß ist die Kullback-Leibler-Divergenz (KL) ([Kullback und Leibler, 1951]). Diese ist definiert durch

$$D_{KL}(q \,||\, p) = \int_x q(x) \log \frac{q(x)}{p(x)} \, \mathrm{d}x = \mathbb{E}_q \left[ \log \frac{q(x)}{p(x)} \right] \tag{2.2}$$

$\mathbb{E}_g \Big[ f(x) \Big]$ beschreibt hier den Erwartungswert von $f(x)$ unter der Funktion $g(x)$. Zu beachten ist dabei, dass die KL nicht symmetrisch ist, das heißt

$$D_{KL}(q \,||\, p) \neq D_{KL}(p \,||\, q) \tag{2.3}$$

Daraus ergibt sich, dass die KL theoretisch kein Distanzmaß ist, da die Symmetrie hierfür eine notwendige Eigenschaft darstellt. Um die beiden Variationen besser unterscheiden zu können wird der linke Term in Gleichung 2.3 daher auch als *Reverse Kullback-Leibler* und der rechte als *Forward Kullback-Leibler* bezeichnet[3].
Im Folgenden werden die Unterschiede der beiden Divergenzen anhand des Terms innerhalb des Integrals untersucht. Das bedeutet es wird dargestellt wie sich die Divergenzen in Bezug auf einen spezifischen Wert für $x$ verhalten. Zudem werden

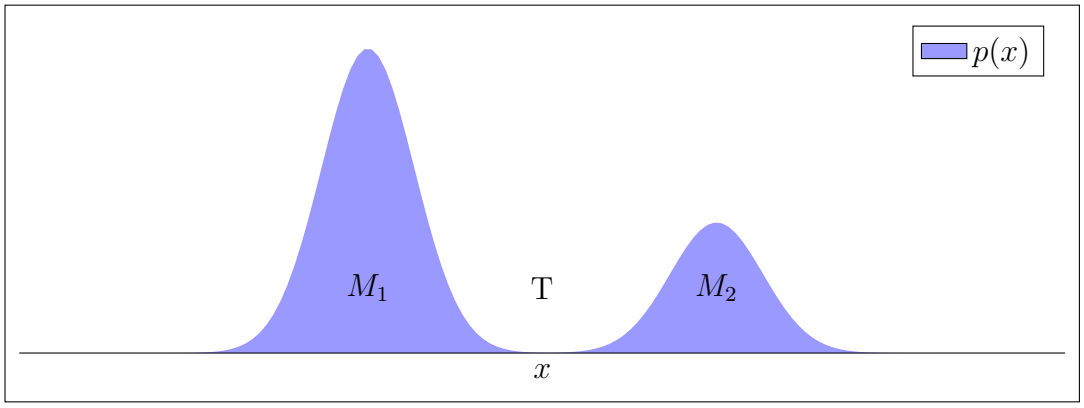



**Abbildung 2.5:** Beispiel einer zu approximierenden Verteilung $p(x)$.
*$M_1$ und $M_2$ sind dabei zwei Normalverteilungen, die unterschiedliche Peaks bilden. T bezeichnet das Tal zwischen diesen.*

[3] In der Literatur wird Reverse Kullback-Leibler auch nur als Kullback-Leibler-Divergenz bezeichnet. Die hier genannten Bezeichner stammen von `https://wiseodd.github.io/techblog/2016/12/21/forward-reverse-kl/`(zuletzt zugegriffen: 25.10.2018) und dienen der besseren Unterscheidung.

die Unterschiede im Bezug auf eine Minimierung anhand des folgenden Beispiels verdeutlicht.
Die Abbildung 2.5 zeigt hier $p$ als eine Mischverteilung aus zwei Normalverteilungen. Als Familie der approximierenden Verteilung $q$ wird in den folgenden Beispielen die Familie der Normalverteilungen genutzt ($q \sim \mathcal{N}(\mu,\sigma)$), da dies für die Anwendung in den Methoden zur Schätzung von Unsicherheiten meist der Fall ist. Dabei ist offensichtlich, dass $q$ die Verteilung $p$ nicht exakt abbilden kann. Allerdings lassen sich anhand dieses Beispiels die Auswirkungen der verwendeten Divergenz gut veranschaulichen.

**Forward KL** Für die Forward KL ergibt sich ein entscheidender Fall für $p(x) \to 0$, denn dann gilt

$$\lim_{p\to 0} f(x) = \lim_{p\to 0} p(x) \log \frac{p(x)}{q(x)} = 0 \tag{2.4}$$

Das bedeutet, dass die Terme, für die $p(x) \to 0$ gilt, nicht in den KL Term mit eingehen. Somit ist es für die Forward KL nicht von Bedeutung, falls sich $q$ und $p$ an diesen Stellen unterscheiden.
Für eine Minimierung würde das bedeuten, dass für alle Bereiche, in denen gilt $p(x) > 0$, vermieden wird für $q(x)$ den Wert 0 zu erreichen. Dieses Verhalten wird auch als *zero avoiding* bezeichnet.
Ein beispielhaftes Ergebnis einer Minimierung der Forward KL zwischen einer festen Verteilung $p(x)$ und einer approximierenden Verteilung $q(x)$ ist in Abbildung 2.6 dargestellt. Dabei bedeckt die sich ergebende Verteilung $q^*(x)$ beide Teile von $p$. Dabei ist der Unterschied im Tal $T$, wie oben beschrieben, nicht von Bedeutung. Problematisch ist hierbei allerdings, dass dem Tal $T$ hier von $q$ eine erheblich größere Wahrscheinlichkeit zugeteilt wird als durch $p$. Die in der Klassifikation verwendete Kreuzentropie kann auf diese Art der KL zurückgeführt werden. Für die Kreuzentropie $H(p,q)$ zwischen der geschätzten Verteilung $q$ (meist Ausgabe einer Softmax-Funktion) und der Zielverteilung $p$ gilt dabei[4][5]:

$$H(p,q) = D_{KL}(p \,||\, q) + H(p) \tag{2.5}$$

[4] `http://www.phonetik.uni-muenchen.de/~reichelu/kurse/stat_lm/script_reichel_stat_lm.pdf`(zuletzt zugegriffen: 25.10.2018)
[5] Eine detaillierte Herleitung der Äquivalenz ist in A.3 zu finden.

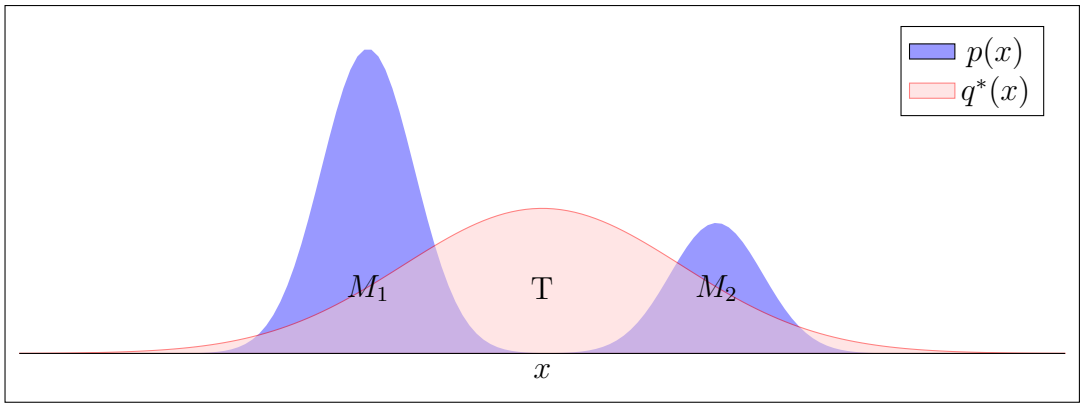



**Abbildung 2.6:** Beispielhaftes Ergebnis für optimales $q^*(x)$ unter Verwendung der Forward KL.
*Die Abbildung zeigt das Ergebnis einer Optimierung der Forward KL zwischen der blauen Verteilung $p(x)$ und $q(x)$. Die sich ergebende Verteilung $q^*(x)$ ist rot markiert, wobei diese die Verteilung $p(x)$ komplett bedeckt. Allerdings teilt $q^*(x)$ dem Tal $T$ eine höhere Wahrscheinlichkeit zu als $p(x)$.*

$H(p)$ beschreibt dabei die Entropie der Verteilung $q$. Da für die Verteilung $p$ gilt:

$$p(c) = \begin{cases} 1, \text{falls } c \text{ die richtige Klasse ist} \\ 0, sonst \end{cases} \tag{2.6}$$

Für die Entropie $H(p)$ bedeutet dies, dass für jedes $p$ gilt $H(p) = 0$. Eine Optimierung der Kreuzentropie entspricht also einer Optimierung der Forward KL. Das zuvor beschriebene Problem der erhöhten Wahrscheinlichkeit des Tals $T$ hat hier allerdings keinen Einfluss. Damit einem Tal eine zu hohe Wahrscheinlichkeit zugeteilt werden kann, muss die Verteilung $p(x)$ multimodal sein. Für die Klassenverteilung trifft dies nicht zu, da im Allgemeinen nur eine Klasse als korrekt gilt und diese Klasse somit den einzigen Modus darstellt.

**Reverse KL** Für die Reverse KL ist genau der umgekehrte Fall wichtig, das heißt wenn $q \to 0$ gilt. Dafür ergibt sich erneut folgender Grenzwert

$$\lim_{q \to 0} f(x) = \lim_{q \to 0} q(x) \log \frac{q(x)}{p(x)} = 0 \tag{2.7}$$

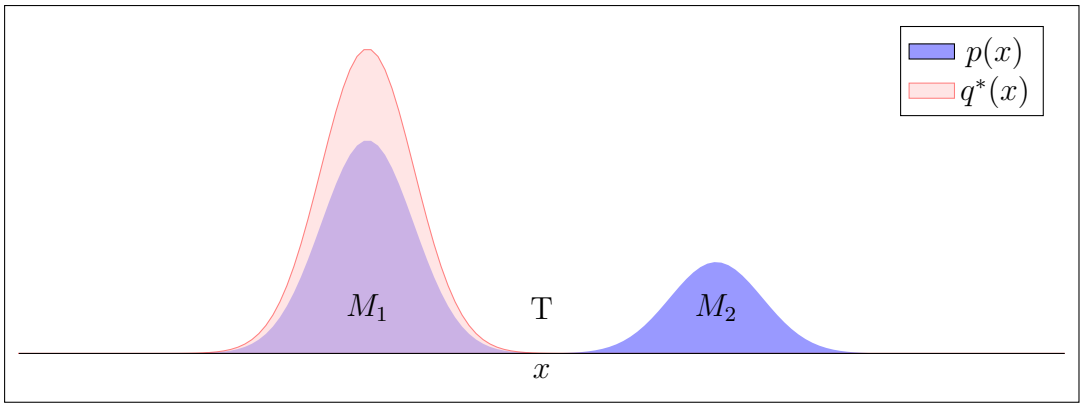



**Abbildung 2.7:** Beispielhaftes Ergebnis für optimales $q^*(x)$ unter Verwendung der Reverse KL.
*$q^*(x)$ konzentriert sich auf einen Peak, da eine optimale Verteilung $q^*(x)$ im Tal $T$ gegen 0 gehen muss.*

Das bedeutet, dass sich $q$ und $p$ in den Bereichen, in denen gilt $q \to 0$, beliebig unterscheiden können.
Die Abbildung 2.7 zeigt nun das Ergebnis der Minimierung unter Verwendung der Reverse KL. Dabei konzentriert sich die Verteilung $q^*$ auf einen der Peaks, da der Wert für $q$ im Tal $T$ auf 0 gedrückt wird. Dieses Verhalten wird deshalb auch *zero forcing* genannt. Der Nachteil hierbei ist, dass der Bereich um $M_2$ von $q$ vernachlässigt wird. Auswirkung hat das besonders auf die Varianz der Verteilung, da in diesem Fall gilt $\mathrm{Var}(q) < \mathrm{Var}(p)$ ist. Da die Varianz der Verteilung letztlich die Unsicherheit darstellt, bedeutet das für Verfahren die nach der Reverse KL optimieren, dass diese dazu tendieren die Unsicherheiten zu unterschätzen.

### 2.4.2 $\alpha$-Divergenz

Bei der $\alpha$-Divergenz [Amari, 1985] handelt es sich um eine Divergenz, für die die KL-Divergenz einen Spezialfall darstellt. Man kann also die $\alpha$-Divergenz als Verallgemeinerung der bisher besprochenen Divergenzen betrachten.

Definiert ist die $\alpha$-Divergenz durch[6]

$$D_\alpha(p \,||\, q) = \frac{1}{\alpha(1-\alpha)}\left(1 - \int p(\theta)^\alpha q(\theta)^{1-\alpha}\,\mathrm{d}\theta\right) \tag{2.8}$$

Für folgende Werte für $\alpha$ ergeben sich dabei die bekannten Variationen der KL-Divergenz[7].

$$D_1(p \,||\, q) = \lim_{\alpha\to 1} D_\alpha(p \,||\, q) = D_{KL}(p \,||\, q) \tag{2.9}$$

$$D_0(p \,||\, q) = \lim_{\alpha\to 0} D_\alpha(p \,||\, q) = D_{KL}(q \,||\, p) \tag{2.10}$$

Der Vorteil dabei ist, dass durch den Parameter $\alpha$ eine Anpassung zwischen *zero forcing* und *zero avoiding* stattfinden kann. Abbildung 2.8 zeigt dafür die resultierenden Verteilungen bei Optimierung mit verschiedenen Werten für $\alpha$. Die beiden Spezialfälle für Variational Inference (VI) und Expectation Propagation (EP) sind markiert. In der praktischen Anwendung hat sich gezeigt, dass für $\alpha = 0.5$ ein besseres Ergebnis erzielt werden kann als für $\alpha \in \{0, 1\}$ [Hernández-Lobato et al., 2016].

[6] Die hier verwendete Formel stellt eine alternative Schreibweise nach [Zhu und Rohwer, 1995] zu der komplexeren Version aus [Amari, 1985] dar.

[7] Ein Beweis für die Grenzwerte ist z.B. in [van Erven und Harremoës, 2014] angegeben.

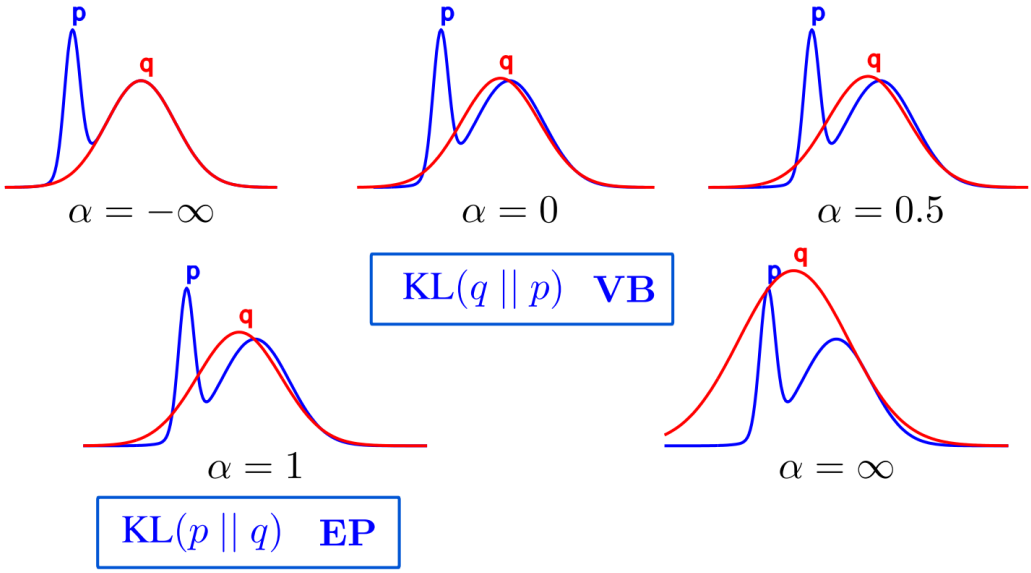



**Abbildung 2.8:** Visualisierung der Auswirkung verschiedener Werte für $\alpha$.
*Zu sehen sind verschiedene beispielhafte Ergebnisse einer Minimierung zwischen $q$ (rote Kurve) und $p$ (blaue Kurve) mit $\alpha \in \{-\infty, 0, 0.5, 1, \infty\}$. Man sieht, dass bei großen Werten für $\alpha$ die Verteilung $q$ immer mehr die gesamte Verteilung $p$ abdeckt, während sich $q$ bei kleinen Werten für $\alpha$ auf einen Modus von $p$ fokussiert. Die Werte* 0 *bzw.* 1 *für $\alpha$ sind als Grenzwerte zu verstehen. Quelle:* [Hernández-Lobato et al., 2016]

# Kapitel 3

# Schätzung von Unsicherheiten

In diesem Abschnitt werden die grundlegenden Prinzipien dargelegt, um Unsicherheiten im Bezug auf Neuronale Netze schätzen zu können. Die verschiedenen Methodiken können dabei, wie in Abbildung 3.1 zu sehen, eingeteilt werden. Im Folgenden werden die verschiedenen Vorgehensweisen beschrieben und es wird jeweils ein Überblick über aktuelle Arbeiten gegeben. Zuletzt werden die aus dem State of the Art gewonnen Erkenntnisse zusammengefasst und deren Bedeutung für diese Arbeit erläutert.

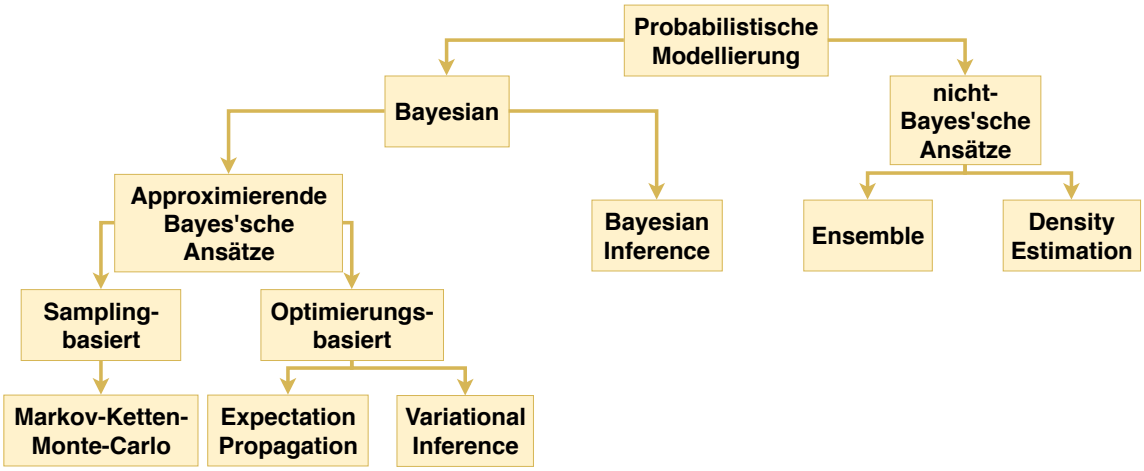



**Abbildung 3.1:** Grundlegende Einteilung der Ansätze in hierarchisch zusammenhängenden Gruppen.

## 3.1 Bayes'scher Ansatz

Der grundsätzliche Unterschied des bayesschen Ansatzes zu der üblichen Herangehensweise im Deep-Learning-Bereich ist, dass statt mit einer Punktschätzung (konkreten Modellparametern) mit Verteilungen über allen möglichen Parametern gearbeitet wird. Dabei lässt sich sowohl das Schlussfolgern anhand von beobachteten Daten als auch das Vorhersagen von Zielgrößen basierend auf einer Eingabe als Formeln aus der bayesschen Statistik darstellen. Diese theoretischen Formulierungen bilden die Grundlage der im Anschluss in Abschnitt 3.2 beschriebenen Verfahren.
Seien $\boldsymbol{D} = \{(\underline{\mathbf{t}}_i, \underline{\mathbf{x}}_i)\}_{i=1}^{M}$ die beobachteten Daten der Tupel mit Eingabe $\underline{\mathbf{x}}_i$ und Teacher $\underline{\mathbf{t}}_i$. Weiterhin bezeichne $\theta$ die Modellparameter, so lässt sich die Inferenz im bayesschen Sinne formulieren als:

$$p(\theta \mid \boldsymbol{D}) = \frac{p(\boldsymbol{D} \mid \theta)p(\theta)}{p(\boldsymbol{D})} = \frac{p(\theta)p(\boldsymbol{D} \mid \theta)}{\int p(\boldsymbol{D} \mid \theta)p(\theta)\,\mathrm{d}\theta} \tag{3.1}$$

$p(\theta)$ beschreibt hierbei den sogenannten *A-priori-Verteilung*. Diese Verteilung beinhaltet Vermutungen oder Vorkenntnisse über die Modellparameter. $p(\theta \mid \boldsymbol{D})$ beschreibt hingegen die sogenannte *A-posteriori-Verteilung*. Diese ist die Schlussfolgerung auf Basis der *A-priori-Verteilung* und der beobachteten Daten $\boldsymbol{D}$. $p(\boldsymbol{D} \mid \theta)$ beschreibt die *Likelihood*-Funktion. Sie beschreibt mit welcher Wahrscheinlichkeit die beobachteten Daten von einem Model mit den gegebenen Parametern $\theta$ erzeugt wurden. Das übliche Training eines Neuronalen Netzes würde einer Maximierung dieser Likelihood-Funktion entsprechen. $p(\boldsymbol{D})$ ist die sogenannte *Evidence*, die beschreibt wie gut ein gewähltes Model zu den Daten passt, unabhängig davon, welche Parameter dieses Model spezifizieren. In der Berechnung der *A-posteriori-Verteilung* wirkt die Evidence als Normierungsfaktor, damit die sich ergebende *A-posteriori-Verteilung* eine eigentliche Verteilung[1] ist.
Um nun mit Hilfe der berechneten *A-posteriori-Verteilung* eine Vorhersage $\underline{\mathbf{y}}^*$ basierend auf einer beliebigen Eingabe $\underline{\mathbf{x}}^*$ zu erhalten kann folgende Formel angewendet werden:

$$p(\underline{\mathbf{y}}^* \mid \underline{\mathbf{x}}^*) = \int p(\underline{\mathbf{y}}^* \mid \underline{\mathbf{x}}^*, \theta, \boldsymbol{D})p(\theta \mid \boldsymbol{D})p(\boldsymbol{D})\,\mathrm{d}\theta = \int p(\underline{\mathbf{y}}^* \mid \underline{\mathbf{x}}^*, \theta)p(\theta \mid \boldsymbol{D})\,\mathrm{d}\theta \tag{3.2}$$

[1] Definition *eigentliche* Verteilung: Das Integral der Verteilungsdichtefunktion ergibt 1.

Zu Beachten ist, dass das Ergebnis $p(\underline{\mathbf{y}}^* | \underline{\mathbf{x}}^*)$ wiederum eine Verteilung darstellt, weshalb diese auch *prädiktive Verteilung* genannt wird. Somit beinhaltet diese Verteilung eine Aussage der Unsicherheit bedingt durch die Unsicherheit in den Parametern beschrieben durch die *A-posteriori-Verteilung*. Das heißt es wird ein gewichteter Mittelwert über den Ausgabeverteilungen jedes möglichen Modells gebildet.
Grundsätzliches Problem an dieser Weise der Inferenz bzw. Prädiktion sind allerdings die benötigten Integrale. Für Problemstellungen, bei denen ein großer Datensatz und ein großes Model verwendet werden, sind diese nicht mehr berechenbar.
Allerdings gibt es Möglichkeiten die benötigten Integrale zu approximieren. Methoden um solche Annäherungen zu erhalten werden im Folgenden beschrieben.

## 3.2 Approximierende bayessche Ansätze

Bei den approximierenden bayesschen Ansätzen geht es prinzipiell darum, die nicht berechenbaren Integrale durch eine Approximation zu ersetzten. Dabei wird meist eine Darstellung der *A-posteriori-Verteilung* gesucht mit Hilfe derer man relativ einfach einzelne Sample generieren kann. Anhand dieser können dann die Integrale durch Monte-Carlo Methoden, das heißt durch einen empirischen Mittelwert und Standardabweichung, approximiert werden.
Um eine Approximation der *A-posteriori-Verteilung* zu erhalten, werden generell zwei verschiedene Herangehensweisen unterschieden.

### 3.2.1 Optimierungs-basiert

Bei der optimierungs-basierten Herangehensweise wird die Approximation der *A-posteriori-Verteilung* als Optimierungsproblem betrachtet. Dabei wird der Unterschied der *A-posteriori-Verteilung* und einer gewählten approximierenden Verteilung minimiert. Dieser wird hierbei gemessen anhand eines Divergenzmaßes, wie beispielsweise die Kullback-Leibler-Divergenz (KL). Die approximierende Verteilung ist hierbei meistens eine parametrisierte Verteilung[2] (z.B. Normalverteilung), sodass die Minimierung anhand dieser Parameter (z.B. Mittelwert und Standardabweichung) erfolgt. Das

[2] Eine Ausnahme stellt dabei zum Beispiel [Liu und Wang, 2016] dar, da hier die Verteilung als Menge von Stichproben dargestellt wird.

bedeutet für die exakte *A-posteriori-Verteilung* $p$, eine approximierende Verteilung $q_\Phi$ mit Parametern $\Phi$ und einem Divergenzmaß $D_f$ wird folgendes Optimierungsproblem gelöst:

$$\Phi^* = \arg\min_{\Phi} D_f(p(\theta) \,||\, q_\Phi(\theta)) \tag{3.3}$$

Die meisten Verfahren lassen sich hier anhand der verwendeten Divergenz in zwei Gruppen einteilen. Verfahren, die die Backward KL-Divergenz verwenden, werden unter dem Begriff Variational Inference (VI) zusammengefasst. Zu dem Bereich der Expectation Propagation (EP) gehören hingegen diejenigen Verfahren, die die Forward KL-Divergenz verwenden.
Bei den Methoden der VI wird dabei nicht direkt auf der oben genannten Formel optimiert. Stattdessen wird der sogenannte *evidence lower bound* für die Optimierung verwendet. Diesen erhält man durch folgende Umformung anhand von Logarithmus-Gesetzen[3].

$$\begin{aligned}
D_{KL}(q(\theta) \,||\, p(\theta \mid \boldsymbol{D})) &= \mathbb{E}_q\left[\log \frac{q(\theta)}{p(\theta \mid \boldsymbol{D})}\right] \\
&= -\,\mathbb{E}_q\left[\log \frac{p(\theta \mid \boldsymbol{D})}{q(\theta)}\right] \\
&= -\,\mathbb{E}_q\left[\log \frac{p(\theta, \boldsymbol{D})}{q(\theta)}\right] + \mathbb{E}_q\Big[\log p(\boldsymbol{D})\Big] \\
&= -\,\mathbb{E}_q\left[\log \frac{p(\theta, \boldsymbol{D})}{q(\theta)}\right] + \log p(\boldsymbol{D}) \\
&= -\,\mathbb{E}_q\left[\log \frac{p(\boldsymbol{D} \mid \theta)p(\theta)}{q(\theta)}\right] + \log p(\boldsymbol{D}) \\
&= -\left(\mathbb{E}_q\left[\log \frac{p(\theta)}{q(\theta)}\right] + \mathbb{E}_q\Big[\log p(\boldsymbol{D} \mid \theta)\Big]\right) + \log p(\boldsymbol{D}) \\
&= D_{KL}(q(\theta) \,||\, p(\theta)) - \mathbb{E}_q\Big[\log p(\boldsymbol{D} \mid \theta)\Big] + \log p(\boldsymbol{D})
\end{aligned} \tag{3.4}$$

$\mathbb{E}_g\Big[f(x)\Big]$ bezeichnet hier wieder den Erwartungswert von $f(x)$ unter der Funktion $g(x)$. Da $\log p(\boldsymbol{D})$ weder von $\theta$ noch von $q$ abhängt kann dieser Summand für die Optimierung vernachlässigt werden. Eine Minimierung von $D_{KL}(q(\theta) \,||\, p(\theta))$ sorgt hierbei dafür, dass die Verteilung $q$ möglichst dicht an der *A-priori-Verteilung* liegt und entspricht somit

[3] Wegen Lesbarkeit wurde statt $q_\Phi(\theta)$ vereinfacht $q(\theta)$ geschrieben.

einer Art Regularisierung. Die Maximierung von $\mathbb{E}_q\Big[\log p(\boldsymbol{D} \mid \theta)\Big]$ entspricht dabei der Maximierung der unter der Verteilung $q$ erwarteten Likelihood der Daten.

**State of the Art** Eine bis heute wichtige Arbeit ist [Neal, 2012], denn in dieser wird erstmals der Zusammenhang von Neuronalen Netzen und Gaußprozessen untersucht. Dieser Vergleich bietet eine wichtige Grundlage für das Verständnis von Neuronalen Netzen in der bayesschen Methodik, sodass auch heutige Arbeiten, wie zum Beispiel [Gal, 2016], aufbauend auf den Ergebnissen dieser Untersuchung argumentieren.
Die wichtigste Literatur im Bereich EP ist [Minka, 2001], in der die grundlegende Vorgehensweise beschrieben wird, und [Minka, 2004], welche eine Erweiterung[4] dies vorherigen Verfahrens beschreibt. Hierbei wird der Likelihood-Term für die Berechnung der *A-posteriori-Verteilung* über den Gewichten faktorisiert, das heißt es wird für jeden Datenpunkt im Trainingsset ein eigener Likelihood-Term betrachtet. Dies bedeutet allerdings, dass der Aufwand dieses Verfahrens stark abhängig von der Anzahl der Daten ist, sowohl in Bezug auf die Rechenzeit als auch auf den Speicherplatz. Als Resultat daraus ist diese Methode für viele Anwendungen nicht praktisch umsetzbar.
Da im Bereich der VI die Optimierung auf dem evidence lower bound geschieht, was den Ressourcenaufwand deutlich verringert, wird diesem Bereich die meiste Aufmerksamkeit zuteil. Eine Übersicht über die Entwicklung und Grundlagen dieser Verfahren bieten dabei [Blei et al., 2016, Jordan et al., 1999, Zhang et al., 2017, Li, 2018].
Ein verbreitetes Verfahren ist das in [Gal und Ghahramani, 2016b] vorgestellte *Monte-Carlo Dropout* (MC-Dropout). Hierbei wird das Training eines Neuronalen Netzes mit Dropout als bayessche Approximation interpretiert. Dabei wird die eigentliche *A-posteriori-Verteilung* durch die Bernoulli-Verteilung, die durch die Dropout-Masken entsteht, approximiert. Um nun eine Schätzung der prädiktiven Verteilung zu erhalten, wird - anders als bei der bisherigen Verwendung von Dropout - auch zur Testzeit Dropout-Masken verwendet. Bei mehrmaliger Berechnung der Netzausgabe unter verschiedenen Masken erhält man nun Sample über deren Standardabweichung die Modellunsicherheit geschätzt werden kann. Die gesamte Unsicherheit in der Modellausgabe setzt sich nun aus dieser geschätzten Modellunsicherheit und einem konstanten Term für die Datenunsicherheit zusammen. Dieser konstante Term wird dabei separat

[4] Dieses Verfahren zeigt parallelen zur Optimierung anhand der $\alpha$-Divergenz.

auf Basis der Trainingsdaten optimiert. Die Unabhängigkeit der Datenunsicherheit von der Eingabe bedeutet dabei, dass hier eine varianzhomogene Datenunsicherheit angenommen wird. Die Schätzung der Modellunsicherheit mittels Dropout in der Testphase hat den entscheidenden Vorteil, dass bei Modellen, die schon zuvor mit Hilfe von Dropout trainiert wurden, keinerlei Änderungen anfallen und man aus diesen sofort eine Schätzung der Unsicherheit erhalten kann. In den Experimenten in [Gal und Ghahramani, 2016b] auf Realdaten wurden dabei Dropout-Raten von 0.05 bzw. 0.005 verwendet. Begründet wurden diese Werte durch die geringe Anzahl der Datenpunkte und der relativ kleinen Modelle (zwei Hidden-Schichten mit jeweils 50 Neuronen). Auch für andere Modelle fällt durch die Verwendung von Monte-Carlo Dropout nur die Dropout-Wahrscheinlichkeiten als zusätzliche Parameter an.
Der zuerst nur auf vollverschalteten Netzen angewandte Ansatz wurde nachfolgend in [Gal und Ghahramani, 2016a] für die in der Bildverarbeitung üblichen Faltungs-Schichten angewandt.
Trotz der weiten Verbreitung und dem Erfolg in der Anwendung, gibt es auch einige Kritik an der theoretischen Grundlagen. Zum Beispiel wurde in [Osband et al., 2018] anhand eines einfachen linearen Beispiels argumentiert, dass das Verhalten der durch MC-Dropout berechneten Unsicherheiten invariant gegenüber der Duplizierung der Trainingsdaten sei. Das heißt die Unsicherheit verändert sich nicht bei Zunahme der Trainingsdaten. Da dieses Verhalten im Widerspruch steht mit der bayesschen Theorie, sollte MC-Dropout eher als Betrachtung eines Ensembles gesehen werden. Eine Lösung dieses Problem stellt das Verfahren *Concrete Dropout*[5] aus [Gal et al., 2017] dar. Hierbei wird der Dropout-Parameter $p$ anhand der KL optimiert. Für eine unendlich große Anzahl an Trainingsdaten gilt hierbei $p \to 0$ und somit geht auch die Modellunsicherheit gegen 0.
Aufbauend auf dem Verfahren von [Gal und Ghahramani, 2016b] wurde in [Li und Gal, 2017] eine Optimierung nicht anhand der KL sondern anhand der $\alpha$-Divergenz durchgeführt. Dafür wurde die zu optimierende Verteilung durch MC-Dropout geschätzt um anhand dieser Samples einen Fehler auf Basis der $\alpha$-Divergenz zu berechnen. Eine der Erkenntnisse dabei war, dass für $\alpha = 0.5$ in der Regel bessere Ergebnisse erzielt werden konnten als mit $\alpha = 0$ (VI) oder $\alpha = 1$ (EP). Eine weitere Arbeit im

[5] Dieses Verfahren wird zwar in den Experimenten dieser Arbeit nicht verwendet, jedoch ist im Anhang A.1.4 eine kurze Diskussion zu diesem Verfahren zu finden.

Zusammenhang mit Dropout ist [Kingma et al., 2015]. In dieser wird überwiegend mit *Gaussian Dropout*, also mit einem Rauschen auf den Gewichten, gearbeitet. Folgend aus den Schlussfolgerungen der Autoren wird dabei eine *A-priori-Verteilung* mit folgender Form verwendet.

$$p(\log(|w_{i,j}|)) \propto c \tag{3.5}$$

Dabei ist $c$ eine Konstante und $w_{i,j}$ das Gewicht zwischen den Neuronen $i$ und $j$. Für genauere Details für die Wahl der *A-priori-Verteilung* sei auf den Anhang von [Kingma et al., 2015] verwiesen. In [Hron et al., 2017] wird genau diese *A-priori-Verteilung* kritisiert, da diese für kein $c$ eine eigentliche *A-priori-Verteilung*[6] bildet. Zwar kann sich durch die Verwendung einer uneigentlichen *A-priori-Verteilung* immer noch eine eigentliche *A-posteriori-Verteilung* ergeben, allerdings zeigen die Autoren, dass das für die logarithmische *A-priori-Verteilung* nicht immer der Fall sein muss. Damit begründen sie, dass *Variational Dropout* im eigentlichen Sinne nicht dem bayesschen Grundwerk entspricht.

### 3.2.2 Sampling-basiert

Sampling-basierte Ansätze basieren auf *Markov-Ketten-Monte-Carlo*-Methoden (Marcov Chain Monte Carlo (MCMC)) [Hastings, 1970, Gelfand und Smith, 1990]. MCMC-Methoden sind hierbei Verfahren um Samples einer gewünschten Verteilung zu generieren. Dabei wird eine Markov-Kette gebildet, deren stationäre Verteilung gleich der zu approximierenden ist. Für die Anwendung als Methode zur Schätzung der Unsicherheiten ist die Zielverteilung die exakte *A-posteriori-Verteilung*. Der allgemeine Ablauf dieser Verfahren ist dabei durch den Pseudocode 3.1 beschrieben. Verschiedene MCMC-Verfahren unterscheiden sich dabei grundsätzlich in der Kandidaten-Funktion $f(\theta)$ und in der Art, auf welche entschieden wird, ob ein Kandidat in die Markov-Kette aufgenommen wird.

Ein entscheidender Nachteil dieser Verfahren ist zum einen der Ressourcen-Verbrauch, da beispielsweise für den Kontext der Neuronalen Netze ein Sample der Markov-Kette einem Set an Parametern für das gesamte Netz entspricht. Bei der Größe heutiger Modelle würde die generierte Markov-Kette enormen Speicherplatz beanspruchen. Zu-

[6] *Eigentliche A-priori-Verteilung:* Das Integral der Verteilung ergibt 1.

**Algorithmus 3.1 :** Pseudocode für den generellen Ablauf von MCMC-Methoden.

**Input :** A-priori-Verteilung $p(\theta)$
Trainingsdaten $D$
Anzahl an Schritten $T$
Kandidaten-Funktion $f(\theta)$

**Output :** Markov-Kette $K$

```
while Länge von K kleiner T do
    Ziehe θ_current aus A-priori-Verteilung p(θ)
    Berechne A-posteriori-Wahrscheinlichkeit p(θ_current | D) = p(θ_current)L(D | θ_current)
    Erzeuge Kandidat für Markov-Kette θ_cand = f(θ_current)
    Berechne A-posteriori-Wahrscheinlichkeit p(θ_cand | D) = p(θ_cand)L(D | θ_cand)
    Entscheide anhand der A-posteriori-Wahrscheinlichkeiten, ob θ_cand der
      Markv-Kette K hinzugefügt wird.
```

dem brauchen diese Verfahren in der Regel sehr lange bis die stationäre Verteilung der Markov-Kette konvergiert, wobei im Vorfeld nicht bekannt ist wie viele Samples die Kette mindestens benötigt[Blei et al., 2016]. Auch ist eine Konvergenz selbst schwer feststellbar und Metriken, welche eine Aussage über die Konvergenz treffen, benötigen meist mehrere Markov-Ketten [Gelman et al., 1992]. Dies bedeutet, dass sich der Rechenaufwand noch einmal vergrößert.

**State of the Art** Trotz der beschränkten Praktikabilität sind diese Verfahren ein interessantes Forschungsgebiet. In [Neal, 2012], beziehungsweise in der vorangegangen Arbeit [Neal, 2011], wird ein Sampling-Algorithmus auf Basis der Hamiltonian Dynamiken vorgestellt und im Zusammenhang mit Neuronalen Netzen diskutiert. Die Hamiltonian Dynamiken stammen dabei eigentlich aus der Physik[7]. In der Praxis ist dieser allerdings schwer anwendbar, da die richtigen Werte für Hyperparameter, wie zum Beispiel die Schrittweite bei der Generierung neuer Samples, gefunden werden müssen. In [Hoffman und Gelman, 2014] wird mit dem No-U-Turn Sampler (NUTS) auf diesem Sampler aufgebaut und ein Vorgehen beschrieben, um speziell für diese Schrittweite automatisch einen passenden Wert zu erhalten. Dabei wird über eine sogenannte *Burn-in* Periode versucht einen geeigneten Wert für die Sprungweite zu

[7] Für eine tiefere Erklärung der Dynamiken und deren Bedeutung für das Verfahren sei auf [Neal, 2011] verwiesen.

finden. Dieses Verfahren wurde auch vom Autor der vorherigen Arbeit [Neal, 2011] kommentiert und unter anderem auch kritisiert[8]. Beispielsweise ist bedenklich, dass die Autoren von [Hoffman und Gelman, 2014] behaupten mit dem vorgestellten Verfahren einen optimal parametrisierten Hamiltonian Sampler zu übertreffen. Dabei versucht ihr Verfahren diese optimalen Hyperparameter selbst zu finden und kann somit theoretisch nicht besser sein als der optimal parametrisierte Sampler. Dennoch stellt die automatische Parametrisierung einen entscheidenden Vorteil dar.
Einen Sonderfall stellt hingegen das Verfahren *Stein Variational Gradient Descend* nach [Liu und Wang, 2016] dar. Hierbei wird in Anlehnung an einen Partikel-Filter die *A-posteriori-Verteilung* angenähert, indem eine feste Anzahl an Samples verändert wird. Durch die Verwendung von einzelnen Samples als Darstellung der approximierenden Verteilung und der Optimierung dieser Sample auf Basis einer Divergenz, stellt dieses Verfahren eine Mischform zwischen den sampling-basierten und den optimierungsbasierten Verfahren dar.

## 3.3 Nicht-bayessche Ansätze

Bei nicht-bayesschen Verfahren beruht die Berechnung der Unsicherheiten nicht auf der zuvor in Abschnitt 3.1 (Seite 18) beschriebenen Theorie. Ein großer Teil dieser Verfahren versucht durch das Bilden eines Ensembles eine Aussage über eine Unsicherheit zu treffen. Diese Unsicherheit wird dabei über den Ausgaben der verschiedenen Modelle des Ensembles gebildet. Dieses Vorgehen kann daher als Sampling der prädiktiven Verteilung durch die Ensemble-Einheiten verstanden werden. Nachteile dieser Verfahren ist dabei, dass die Modelle des Ensembles eine gewisse Diversität aufweisen müssen und separat trainiert werden müssen. Für die Anwendung in heutigen Problemstellungen stellt dies ein Problem dar, da die verwendeten Modelle sehr groß werden können.
Ein weiterer Ansatz ist es, die Ausgabe eines Modells direkt als parametrisierte Verteilung zu definieren. Modelle, die dieser Methodik folgen werden Mixture Density Networks (MDN) [Bishop, 1994] genannt. Dabei werden statt einer Punktschätzung $\underline{\mathbf{y}}$ die Parameter der Verteilung $P(\underline{\mathbf{y}}|\underline{\mathbf{x}},\theta)$ geschätzt. Zum Beispiel kann anhand eines Mittelwerts $\mu$ und einer Varianz $\sigma$ die prädiktive Verteilung anhand einer Normal-

[8] https://radfordneal.wordpress.com/2012/01/27/evaluation-of-nuts-more-comments-on-the-paper-by-hoffman-and-gelman/(zuletzt zugegriffen: 25.10.2018)

verteilung approximiert werden. Durch diesen Ansatz lassen sich auch multimodale Verteilungen direkt schätzen indem Parameter für mehrere Verteilungen gleichzeitig geschätzt werden. Die Abbildung 3.2 verdeutlicht die Änderungen eines herkömmlichen Models. Die üblicherweise verwendeten Fehlerfunktionen müssen allerdings an die neue Ausgabe angepasst werden, so müsste zum Beispiel statt des *Mean-Squared-Error* direkt die *Gaussian-Likelihood* verwendet werden.
Nachteil dieses Ansatzes ist allerdings, dass hierbei nur die Unsicherheit modelliert wird, die in den Daten selbst repräsentiert wird. Die Unsicherheit der geschätzten Modellparameter wird hier vernachlässigt, da nur mit einer Punktschätzung der Parameter gearbeitet wird. Der Vorteil dagegen besteht darin, dass die Datenunsicherheit auch in varianzheterogener Form geschätzt werden kann.

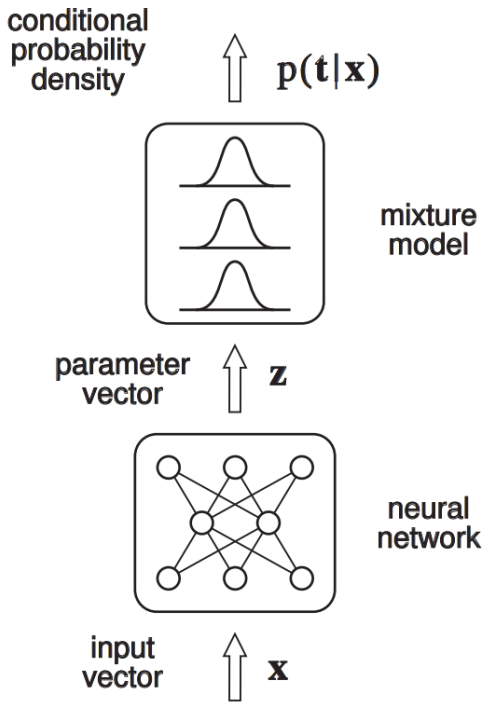



**Abbildung 3.2:** Veranschaulichung des Prinzips von Mixture-Density-Models.
*Die Abbildung zeigt dabei schematisch wie anhand einer Eingabe* $\underline{\mathbf{x}}$ *durch ein Neuronales Netz die Parameter* $\underline{\mathbf{z}}$ *einer parametrisierten Verteilungen (z.B. Normalverteilung) berechnet wird. Die resultierenden Verteilungen beschreiben in der Summe die Verteilung der Zielwerte* $\underline{\mathbf{t}}$ *bedingt durch die Eingabe* $\underline{\mathbf{x}}$*. Quelle:* [Bishop, 1994]

**State of the Art** Im Sinne der MDN wird in [Kendall und Gal, 2017] durch einen zusätzlichen Output der Modelle die Datenunsicherheit direkt geschätzt. Zusätzlich wurde die Modellunsicherheit separat geschätzt, um eine Trennung der beiden Unsicherheiten zu ermöglichen. Die Modellunsicherheit (epistemic uncertainty) wurde durch das in [Gal und Ghahramani, 2016b] beschriebene Verfahren geschätzt. Die dabei angepasste Fehlerfunktion hat die Form:

$$L = \frac{1}{2}\hat{\sigma}^{-2}(y_{true} - \hat{y})^2 + \frac{1}{2}\log\hat{\sigma}^2 \tag{3.6}$$

Wobei $\hat{y}$ die bisherige Schätzung darstellt und $\hat{\sigma}$ die Schätzung der Datenunsicherheit ist[9]. Diese Fehler-Funktion entspricht dabei bis auf einen konstanten Faktor der negativen Log-Likelihood

$$LL(x) = \mathcal{N}(y_{true}; \hat{y}, \hat{\sigma}) = \frac{1}{2}\hat{\sigma}^{-2}(y_{true} - \hat{y})^2 + \frac{1}{2}\log\hat{\sigma}^2 + \frac{1}{2}\log 2\pi \tag{3.7}$$

In der Anwendung für die Regression verwenden die Autoren allerdings die Laplace-Variante der Log-Likelihood

$$LL_{Laplace}(x) = \frac{1}{\hat{\sigma}}|y_{true} - \hat{y}| + \log\hat{\sigma} \tag{3.8}$$

da diese laut den Autoren für Regressionsprobleme bessere Ergebnisse liefert, wobei diese Aussagen nicht konkret belegt wurde, sondern nur auf den bisherigen Erfahrungen der Autoren zu basieren scheint. Für einen groben Vergleich der beiden Likelihood-Funktionen zeigt die Abbildung 3.3, neben den beiden Verteilungsfunktionen, den Verlauf der negativen Log-Likelihood basierend auf der Laplace- bzw. Gauß-Verteilung. Anhand der Abbildung ist allerdings erkennbar, dass die beiden Parameter $\sigma$ und $b$ der beiden Verteilungen, welche die Stärke der Streuung angeben nicht direkt vergleichbar sind. Erkennbar ist dies an den unterschiedlichen Minima der negativen Likelihood für den Streuungsparameter bei standardnormalverteilten Daten. Es muss daher darauf geachtet werden, dass der numerische Wert der geschätzten Datenunsicherheit richtig interpretiert wird.

Das Verfahren wurde für die Semantische Segmentierung und für die Tiefenschätzung als Repräsentation eines Klassifikations- bzw. Regressionsproblems angewandt. Die

[9] Damit die geschätzte Varianz stets positiv ist wird in der Praxis $\log\hat{\sigma}^2$ statt $\hat{\sigma}^2$ optimiert.

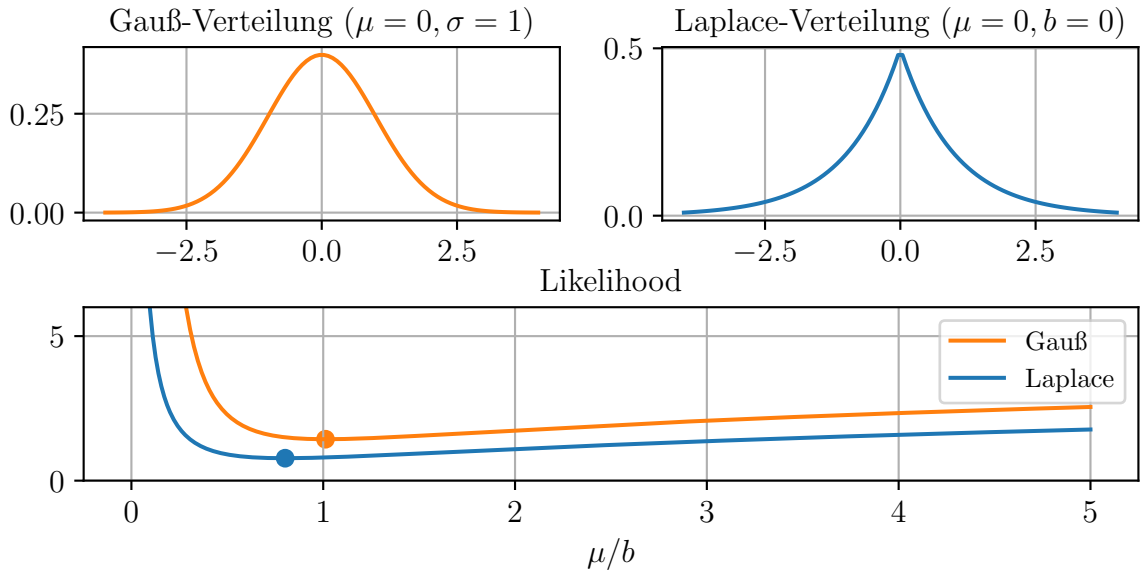



**Abbildung 3.3:** Vergleich zwischen Laplace- und Gauß-Verteilung.
*Die Abbildung zeigt neben der Verteilungsfunktionen einen Vergleich zwischen dem Verlauf der negativen Log-Likelihood. Die zugrundeliegenden Daten sind dabei standardnormalverteilt. Das Minimum der negativen Likelihood ist jeweils durch einen Punkt gekennzeichnet.*

Abbildung 3.4 zeigt eine Visualisierung der gewonnenen Unsicherheiten. Die Autoren heben hervor, dass anhand dieser Abbildung zu erkennen ist, dass die Datenunsicherheit überwiegend an den Grenzen der Bereiche geschätzt wird. Die sei ein Zeichen für den Erfolg der Schätzung, da sich die Labelinformationen in diesen Bereichen über den Datensatz hinweg widersprechen können und somit als Datenunsicherheit zu klassifizieren ist.

Eine wichtige Arbeit im Zusammenhang mit dem Ensemble-Ansatz ist [Lakshminarayanan et al., 2017]. In dieser werden verschiedene Techniken untersucht, um eine Diversität der einzelnen Modelle zu erreichen.

In [Osband et al., 2018] wird die Qualität bestehender Methoden zur Schätzung von Unsicherheiten im Kontext von sequentiellen Entscheidungsproblemen näher untersucht. Als Lösungsansatz, der dort beschriebene Nachteile ausgleichen soll, wird das Prinzip einer *A-priori-Verteilung* in das Training eines Ensembles integriert. Dabei wird eine Verteilung über Funktionen $P$ definiert, welche als Repräsentation des Vorwissens dient. Jedem Model des Ensembles wird nun eine sogenannte *prior function* $p \sim P$ zugeordnet.

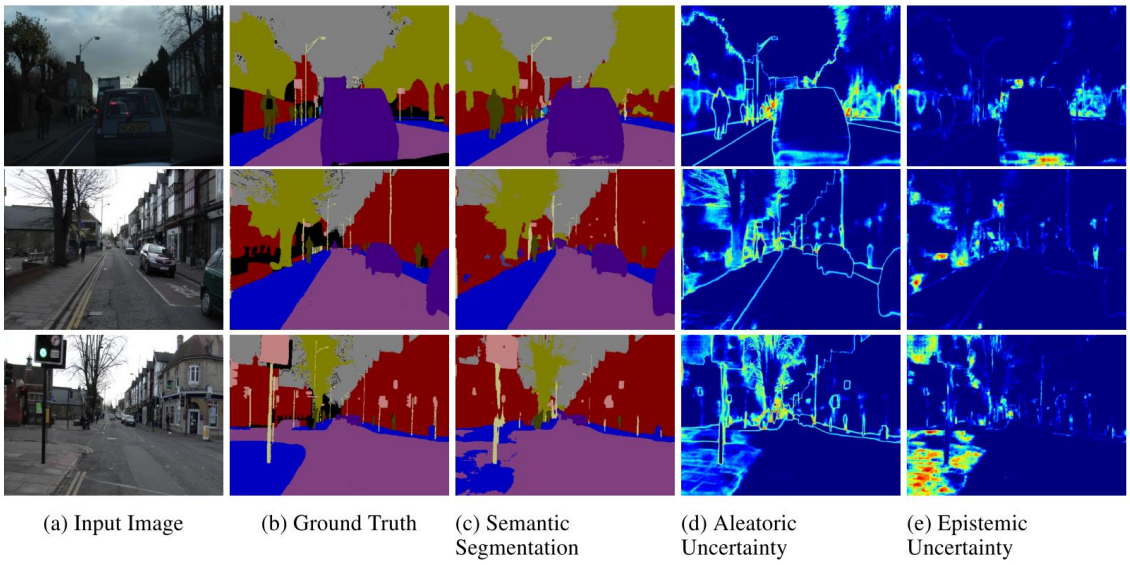



**Abbildung 3.4:** Beispiel einer Sematischen Segmentierung mit getrennter Schätzung für Daten- bzw. Modellunsicherheit.
*(a) Eingabe für das Model, (b) Label-Daten als Ground Truth, (c) Semantische Segmentierung des Models, (d) Geschätzte Datenunsicherheit, (e) Geschätzte Modellunsicherheit. Datenunsicherheit überwiegend an Klassengrenzen geschätzt, da sich hier Labels oft widersprechen können. Quelle:* [Kendall und Gal, 2017]

Bei der Optimierung der einzelnen Modelle wird nun die jeweilige prior function $p$ auf die Ausgabe des Modells addiert. Zur Veranschaulichung des Verfahrens beschreibt der Pseudocode 3.2 den generellen Ablauf. Die Funktionen $p_k$ werden dabei durch zufällig initialisierte Netze dargestellt. Eine genauere Untersuchung der möglichen Wahl für die Verteilung $P$ lassen die Autoren allerdings offen.
Anhand dieser Arbeit zeigt sich, dass es selbst für die Verfahren, welche nicht auf Basis der bayesschen Theorie arbeiten, von Vorteil sein kann Vorwissen in Form eines Priors in den Inferenzprozess zu integrieren, um geeignete Unsicherheiten extrahieren zu können. Allerdings bedeutet das auch, dass Verfahren die auf der bayesschen Theorie aufsetzten erfolgversprechend sind.
Bei Modellen, die bei Klassifikationsproblemem verwendet werden, wird meist in der Ausgabe-Schicht eine Softmax-Aktivierung verwendet. Diese skaliert die Ausgabe so, dass sie in der Summe eins ergibt. Trotz, dass die dadurch entstehende Verteilung keinen theoretischen Zusammenhang mit der prädiktiven Verteilung besitzt, wird sie oft

**Algorithmus 3.2 :** Pseudocode zum allgemeinen Ablauf des Verfahrens aus [Osband et al., 2018]. Dieser Pseudocode stammt dabei direkt aus der Arbeit und wurde lediglich übersetzt.

**Input :** Daten $\boldsymbol{D}$
Fehlerfunktion $L(\underline{\mathbf{y}},\boldsymbol{D})$
Model $\underline{\mathbf{y}} = f_\theta(\underline{\mathbf{x}})$ spezifiziert durch Modellparameter $\theta$
Größe des Ensembles $K$
Daten-Funktion $data_noise(\boldsymbol{D})$
Verteilung über prior functions $P(p)$

**Output :** Ensemble $\{f_{\theta_k} + p_k\}_{k=1}^K$

```
for k = 1...K do
    Initialisiere θ_k zufällig
    Erzeuge D_k = data_noise(D)          // Zum Beispiel Bootstrapping
    Erzeuge p_k ~ P
    Optimiere L(f_θk + p_k, D_k) unter den Modellparametern θ_k
```

herangezogen, um eine Aussage über die Konfidenz oder Unsicherheit eines Modelles zu machen.

In [Guo et al., 2017] wird gezeigt, dass neuere Modelle (ResNet) zwar immer besser werden, allerdings sind diese immer schlechter kalibriert. Die Abbildung 3.5 zeigt einen Vergleich zwischen LeNet aus dem Jahr 1998 und dem aktuelleren ResNet aus dem Jahr 2016. Die Abbildung veranschaulicht, dass das neuere ResNet zwar einen geringen Fehler erzielt aber die Unsicherheit der Klassifikation stark unterschätzt[10].

Da diese Unsicherheiten dabei direkt anhand der Softmax-Ausgabe abgelesen wurden bedeutet das, dass diese keine geeigneten Konfidenz- oder Unsicherheitsangaben darstellt. Um diesem Problem entgegenzuwirken schlagen die Autoren eine Kalibrierung der Softmax-Ausgabe vor. Dabei nennen und vergleichen sie verschiedene Methoden, wie zum Beispiel Histogramm Binning oder Temperature Scaling. Bei diesen Methoden wird die Ausgabe der Sofmax-Funktion anhand eines zusätzlichen Datensatzes angepasst, sodass diese auf diesem zu kalibrieren Konfidenzwerten führt sind. Untersucht werden diese Methoden dabei auf verschiedene Datensätze aus der Bild- und Textklassifikation. Ein großer Vorteil dieser Vorgehensweise ist der geringe Aufwand, um kalibrierte Konfidenzwerte zu erhalten. Die Optimierung des Kalierungsfaktors beim Tempearture

[10] Das Prinzip der verwendeten Kalibrierungskurven wird in 3.4.1 erläutert

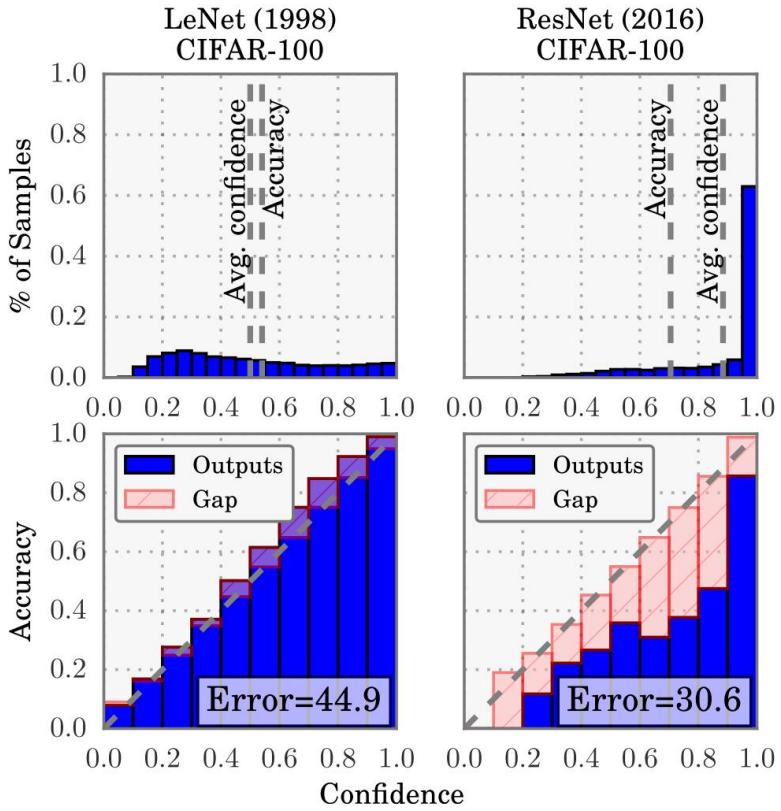



**Abbildung 3.5:** Vergleich der Kalibrierung früherer Modelle mit heutigen.
*Die Abbildung zeigt einen Vergleich der Kalibrierung von LeNet (1998) und ResNet (2016) auf dem CIFAR-100 Datensatz. Zu erkennen ist dabei, dass LeNet eine gut kalibrierte Schätzung der Confidence ermöglichte. ResNet dagegen unterschätzt seine Unsicherheit stark. Quelle:* [Guo et al., 2017]

Scaling ist bei Verwendung üblicher Gradientenoptimierer innerhalb weniger Iterationen abgeschlossen[11]. Im Vergleich mit den anderen Kalibrierungs-Methoden besitzt dieses Verfahren damit den geringsten Rechenaufwand.
Ähnlich zu diesem Prinzip wurde in [Neumann et al., 2018] vorgeschlagen, die Temperatur, anhand derer die Softmax-Ausgabe skaliert wird, während des Trainingsprozesses zu optimieren. Zudem wird die Temperatur in Abhängigkeit der Eingabe berechnet. Somit bietet dieses Verfahren, ähnlich zu [Kendall und Gal, 2017], eine Möglichkeit, varianzheterogene Datenunsicherheit darzustellen.

[11] Bei Verwendung des Optimierungsverfahrens LM-BFGS ist die Optimierung des Skalierungsfaktors nach zehn Iterationen abgeschlossen.

## 3.4 Bewertung von Unsicherheiten

Um die geschätzten Unsicherheiten bewerten zu können, sind Werkzeuge nötig, durch die eine qualitative Aussage über diese Schätzungen gemacht werden kann. Grundsätzlich wird dabei von den Unsicherheiten erwartet, dass diese in Zusammenhang mit dem Fehler des Modells stehen. Das bedeutet, dass ein Modell seine eigenen Fehler durch eine entsprechend hohe Unsicherheit erklären können sollte.
Im folgenden Kapitel werden nun Werkzeuge, welche eine Bedeutung für diese Masterarbeit haben, vorgestellt und erläutert. Dabei handelt es sich um die Kalibrierungskurven und die Likelihood, welche auch im State of the Art Verwendung finden.

### 3.4.1 Kalibrierungskurve

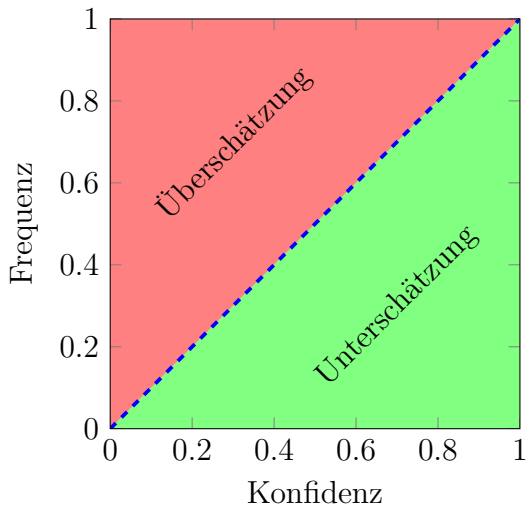



**Abbildung 3.6:** Aufbau eines Kalibrierungs-Plots.
*In einem Kalibrierungsplot wird entlang der x-Achse die Unsicherheit bzw. Konfidenz in Prozent und entlang der y-Achse eine Frequenz oder Häufigkeit angegeben. Die Bedeutung der Unsicherheit und die Art über die die Frequenz berechnet werden, unterscheiden sich dabei für Klassifikation und Regression. Gestrichelte Linie stellt die Idealkurve dar.*

Kalibrierungskurven[12] sind ein Mittel um eine qualitative Aussage über geschätzte Unsicherheiten in Form von Standardabweichungen zu geben. Dabei wird überprüft

[12] Im Englischen auch *reliability diagram* genannt.

inwieweit die Unsicherheiten die Fehler des Modells erklären, das bedeutet wenn das Modell eine hohe Unsicherheit angibt, sollte es in diesen Fällen auch einen erhöhten Fehler machen und falls das Modell eine sehr kleine Unsicherheit angibt sollte es vergleichbar kleine Fehler machen. Der Allgemeine Aufbau ist in Abbildung 3.6 zu sehen[13].
Die Berechnung dieser Kurven unterscheidet sich für Klassifikation und Regression wobei diese jeweils über einen Datensatz $D$ berechnet werden. Beide Formen werden im Folgenden erläutert.

### Klassifikation

Für die Klassifikation sieht die Berechnung der Kalibrierungskurve aufgrund der oft verwendeten Softmax-Ausgabe anders aus als bei der Regression. Der Unterschied ergibt sich dabei durch die diskrete Verteilung der Klassenwahrscheinlichkeiten (Softmax-Ausgabe) im Gegensatz zu der kontinuierlichen Verteilung bei der Regression. Um einen Punkt der Kalibrierungskurve $(p, \cdot)$ zu berechnen muss wie folgt vorgegangen werden. Zuerst wird die Ausgabe des Modells über einem Datensatz berechnet, der keine Punkte enthält, welche für das Training verwendet wurden. Nun werden alle Fälle gesammelt, bei denen eine Klasse $c$ die Wahrscheinlichkeit $p$ zugeteilt bekommen hat. Danach kann der Anteil $z_p$ dieser Fälle berechnet werden, bei denen die Klasse $c$ auch der richtigen Klasse entsprochen hat. Bei einem perfekt kalibriertem Modell würde nun gelten $z_p = p$ für alle $p \in [0,1]$.
Das bedeutet zum Beispiel, dass in 20% der Fälle, in denen das Modell einer Klasse eine Wahrscheinlichkeit von 20% zugeteilt hat, diese Klasse auch der Ground Truth entsprechen sollte.
Da allerdings dabei jeder mögliche Wert für $p$ berücksichtigt werden müsste und es Klassenwahrscheinlichkeiten geben wird, welche nicht vorkommen werden, wird das Intervall $[0,1]$ für $p$ in Teilintervalle eingeteilt. Das bedeutet, dass anstatt beliebiger Werte für $p$ nun Bins betrachtet werden, deren Breite meist konstant ist.
Im Gegensatz zur Regression können im Falle der Klassifikation weitere Metriken verwendet werden, um die Qualität der Unsicherheiten zu bewerten. Dabei werden die geschätzten Unsicherheiten meist mit der erreichten Accuracy verglichen. Intuitiv

[13] Über- bzw. Unterschätzung sind dabei in Bezug auf die Unsicherheit zu verstehen. Im Kontext der Konfidenz tauschen die beiden Bereiche die Plätze.

**Algorithmus 3.3 :** Pseudocode zur Berechnung der Kalibrierungskurve für die Klassifikation.

**Input** : Modell $f(x) = (y, p)$ mit prädizierter Klasse $y$ und der dazugehörigen Wahrscheinlichkeit $p$
Datensatz $\boldsymbol{D} = \{(t_i, x_i)\}_{i=1}^{N}$
Anzahl der Bins $M$

**Output** : Punkte der Kalibrierungskurve $P = \{(i, z_i)\}_{i=1}^{M})$

```
1 P ← {}
2 for m ← 1 to M do
      /* Sammle Indices aller Datenpunkte bei denen p_i im Bin liegt */
3     B_m = {i | p_i ∈ ((m−1)/M, m/M]}
      /* Berechne Anteil der Datenpunkte bei denen Prädiktion richtig lag
         (Accuracy) */
4     z_m = (1/|B_m|) Σ_{i∈B_m} δ(y_i = t_i)
5     P ⋃ {(m, z_m)}                    // Füge neuen Punkt zu P hinzu
```

kann dabei erwartet werden, dass ein Modell welches auf einem Datensatz eine mittlere Unsicherheit von 20% hat auch 20% dieses Datensatzes richtig klassifizieren wird, falls die Unsicherheiten gut kalibriert sind.
In [Guo et al., 2017] werden beispielsweise zwei Metriken definiert. Für diese wird eine Accuracy und eine Konfidenz über den einzelnen Indexsets $B_m$ nach folgenden Formeln berechnet:

$$\begin{aligned} \mathrm{acc}(B_m) &= \frac{1}{|B_m|} \sum_{i \in B_m} \delta(y_i = t_i) \\ \mathrm{conf}(B_m) &= \frac{1}{|B_m|} \sum_{i \in B_m} p_i \end{aligned} \tag{3.9}$$

Der Expected Calibration Error (ECE) beschreibt den gewichteten Kalibrierungsfehler über alle Bins und ist definiert durch:

$$ECE = \sum_{m=1}^{M} \frac{|B_m|}{N} \Big| \mathrm{acc}(B_m) - \mathrm{conf}(B_m) \Big| \tag{3.10}$$

Der Maximum Calibration Error (MCE) beschreibt den größten Kalibrierungsfehler über alle Bins und ist definiert durch:

$$MCE = \max_{m \in \{1,\dots,M\}} \Big| \operatorname{acc}(B_m) - \operatorname{conf}(B_m) \Big| \tag{3.11}$$

Diese Metriken bieten einen skalaren Wert anhand dessen die Kalibrierung von Modellen verglichen werden kann, ohne visuelle Vergleiche der Kalibrierungskurven anzustellen.

### Regression

Folgende Erklärungen basieren auf einem 1D-Problem, d.h. $\boldsymbol{D} = \{(x_i, t_i)\}_{i=1}^{N}$ mit Eingabe $x_i \in \mathbb{R}$ und Zielwert $t_i \in \mathbb{R}$. Zudem sind die Ausgaben des Modells $y_i$ und $\sigma_i$ für eine Eingabe $x_i$ gegeben. Dabei ist $y_i$ der prädizierte Wert für $t_i$ und $\sigma_i$ die geschätzte Unsicherheit. Die Bedeutung von $\sigma_i$ ist abhängig von der angenommen Verteilung der Daten. In den meisten Fällen, wie auch in den folgenden Erklärungen, wird hier eine Normalverteilung angenommen[14] und somit wird $\sigma_i$ als Standardabweichung interpretiert. Somit kann für jede Eingabe nun eine Normalverteilung definiert werden durch $\mathcal{N}_i(0, \sigma_i)$. Für jeden Wert $p$ entlang der x-Achse des Kalibrierungs-Plots kann nun anhand dieser Normalverteilungen ein Intervall $[-k_i\sigma_i, k_i\sigma_i]$ bestimmt werden, für das gilt:

$$\int_{-k_i\sigma_i}^{k_i\sigma_i} \mathcal{N}(y|0, \sigma_i)\, \mathrm{d}y = p \tag{3.12}$$

Das heißt die Fläche über diesem Intervall muss gleich $p$ sein. Die Abbildung 3.7 zeigt ein Beispiel einer Standard-Normalverteilung, bei der die Fläche für $p = 68.27\%$ blau markiert ist. Das benötigte $k$ kann hierbei berechnet werden durch:

$$k_i = \operatorname{erf}^{-1}(p) \cdot \sqrt{2} \tag{3.13}$$

Dabei bezeichnet $\operatorname{erf}^{-1}(p)$ die Inverse der gaußschen Fehlerfunktion $\operatorname{erf}(x)$, welche definiert ist durch:

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, \mathrm{d}t \tag{3.14}$$

[14] Die Laplace-Verteilung stellt eine weitere verwendete Verteilung dar.

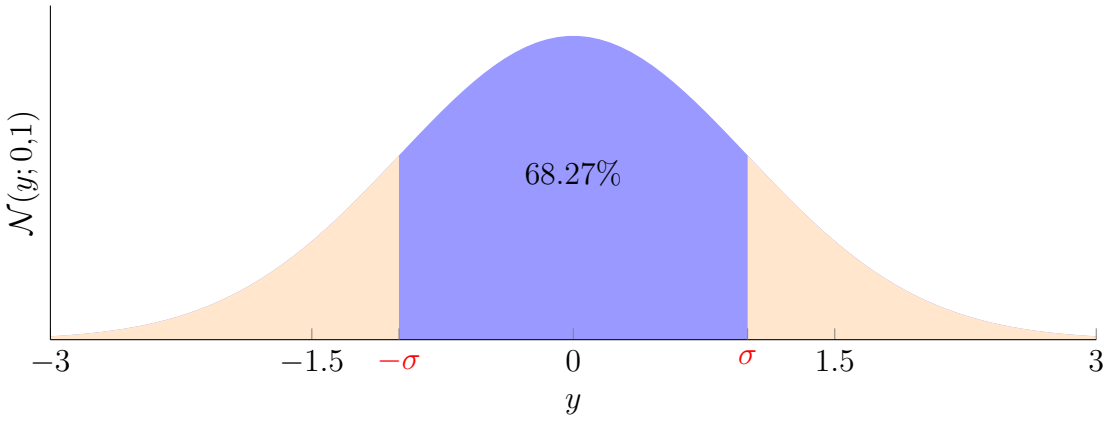



**Abbildung 3.7:** Beispiel des zu berechnenden Intervalls.
*Zu sehen ist die durch die Vorhersage $\hat{y}$ und die Unsicherheit $\sigma$ aufgespannte Normal-Verteilung. Für dieses Beispiel wird der Punkt $(\cdot, 0.6827)$ der Kalibrierungskurve berechnet. Die blaue Fläche markiert dafür 68.27% der Normal-Verteilung. Die Grenzen des gesuchten Intervalls ergeben sich hier zu $[-\sigma,\sigma]$ $(k = 1)$ und sind rot markiert.*

Nachdem für jede Eingabe $x_i$ diese Intervalle bestimmt sind, wird der Anteil der Datenpunkte $z$ für die der Fehler $e_i = y_i - \hat{y}_i$ in das jeweilige Intervall fallen, berechnet. Das heißt $z$ ergibt sich zu:

$$z = \frac{1}{N} \sum_{i=1}^{N} \delta_i \tag{3.15}$$

mit:

$$\delta_i = \begin{cases} 1, & e_i \in [-k_i\sigma_i, k_i\sigma_i] \\ 0, & sonst \end{cases} \tag{3.16}$$

Somit erhält man den Punkt $(p, z)$ der Kalibrierungskurve. In Worte gefasst beschreibt die Kalibrierungskurve für die Regression somit für jedes Konfidenzintervall, in wie vielen Fällen der Zielwert in das Intervall fällt. Der Pseudocode 3.4 beschreibt noch einmal den Ablauf bei der Berechnung der Kalibrierungskurve.
Um verschiedene Kalibrierungskurven vergleichen zu können, kann ein Fehler zwischen der Kurve und der Ideal-Kurve berechnet werden. Das verwendete Fehlermaß ist dabei meistens der *Mean Squared Error*.

**Algorithmus 3.4 :** Pseudocode für die Berechnung der Kalibrierungskurve für die Regression.

**Input** : Prädiktion der Mittelwerte $\underline{\mathbf{y}}$
Prädiktion der Unsicherheiten $\underline{\sigma}$
Ground Truth Daten $\underline{\mathbf{t}}$
Anzahl an diskreten Punkten der Kurve $M$
Anzahl der Daten $N$

**Output** : Punkte der Kalibrierungskurve $P$

```
1 P ← {}                                  // Menge der Punkte der Kalibrierungskurve
2 for p ← 0 to 1 by 1/M do
3     k = erf^{-1}(p)√2                   // Bestimme k für die Berechnung des Intervalls
      /* Zähle Anzahl z der Fälle in denen der Fehler im Interval liegt  */
4     z ← 0
5     for i ← 0 to N by 1 do
6         e_i = t_i − y_i                 // Berechne Fehler
7         if e_i ∈ [−kσ_i, kσ_i] then
8             z = z + 1
      /* Füge neuen Punkt der Menge P hinzu                              */
9     P = P ⋃ {(p, z/N)}
```

**Verhalten bei Fehlern des Modells** Um die Kalibrierungskurven effektiv bewerten zu können, muss das Verhalten dieser unter verschiedenen Situationen untersucht werden. Basierend auf diesem Wissen können mögliche Ursachen einer Misskalibrierung beschrieben werden.

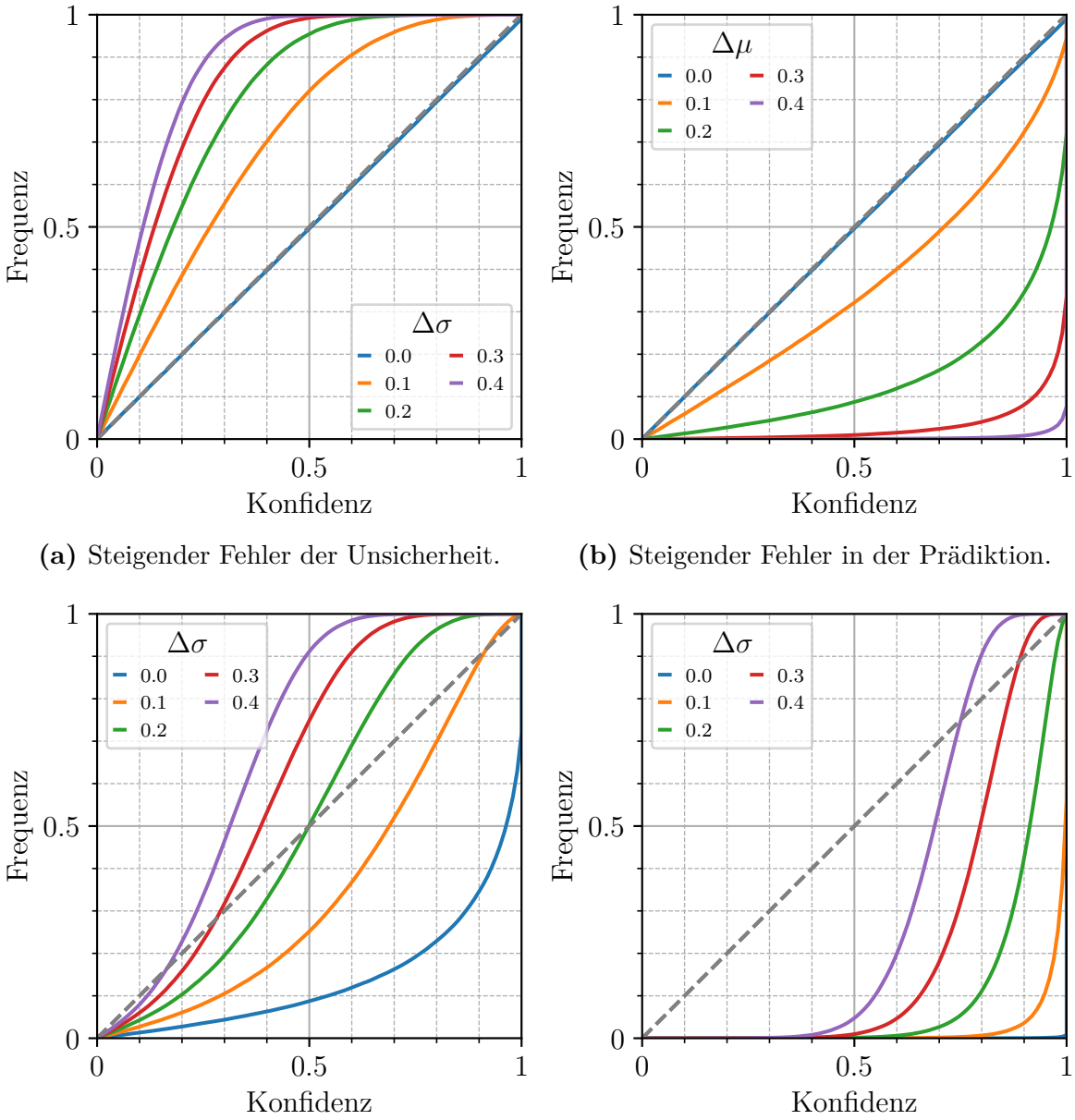
Die dabei verwendeten Daten stammen aus dem varianzhomogenen Datensatz, welcher einen der Datensätze darstellt, die in dieser Masterarbeit verwendet wurden und in Abschnitt 4.1 (Seite 45) beschrieben wird. Dieser Datensatz wurde dabei durch folgende Funktion generiert:

$$\begin{aligned} f(x) &= \cos(x) + \epsilon \\ \epsilon &\sim \mathcal{N}(0{,}0.1) \end{aligned} \tag{3.17}$$

Das bedeutet ein ideales Model hätte die Ausgabe $y = (\mu(x), \sigma(x))$ mit der Prädiktion des Mittelwertes $\mu(x) = \cos(x)$ und der Unsicherheit $\sigma(x) = 0.1$.

In der Abbildung 3.8 sind nun die Kalibrierungskurven des idealen Modells, dessen

Ausgabe $y(x) = (\mu(x), \sigma(x))$ durch Fehlerterme $\Delta\mu$ bzw. $\Delta\sigma$ verfälscht wurden, dargestellt.
Dies stellt zwar eine starke Vereinfachung dar, da ein Modell nie für jeden Datenpunkt den gleichen Fehler machen wird, jedoch lassen sich anhand dieses vereinfachten Sachverhaltes wichtige Eigenschaften der Kalibrierungskurven ablesen.
Anhand der Abbildungen 3.8a und 3.8b sind die Auswirkungen eines Fehlers im Mittelwert bzw. in der Unsicherheit zu sehen. Erkennbar ist, dass sich beide Fehlerarten in einer resultierenden einfachen Unter- bzw. Überschätzung äußert, da die Kalibierungskurven stets unter bzw. über der Diagonalen liegen. Es stellt sich nun die Frage, ob ein Modell, welches einen konstanten Fehler in der Prädiktion des Mittelwertes macht, den dadurch resultierenden Kalibrierungsfehler durch eine konstante Änderung der Unsicherheiten beseitigen kann.
Die beiden unteren Abbildungen in 3.8 zeigen hierfür die Kalibrierungskurven eines Modelles mit konstantem Fehler $\Delta\mu = 0.2$ (3.8c) bzw. mit $\Delta\mu = 0.5$ (3.8d) und variierender Änderung der Unsicherheit $\Delta\sigma$. Zu erkennen ist, dass es keine konstante Änderung $\Delta\sigma$ gibt, für die die Kalibrierungskurve auf die Idealkurve gebracht werden kann. Mit zunehmender Unsicherheit nähert sich die Kalibrierungskurve einer Stufenfunktion. Für den Fall, dass eine solche Kurve einen Schnittpunkt mit der Diagonalen besitzt, würde dies bedeuten, dass das Modell geringe Unsicherheiten unterschätzen und größere Unsicherheiten überschätzen würde.




**(a)** Steigender Fehler der Unsicherheit.

**(b)** Steigender Fehler in der Prädiktion.

**(c)** Konstanter Fehler $\Delta\mu = 0.2$.

**(d)** Konstanter Fehler $\Delta\mu = 0.5$.

**Abbildung 3.8:** Verlauf der Kalibrierungskurve bei Fehlern in der Unsicherheit oder der Prädiktion.

*(a) Steigender Fehler $\Delta\mu$ des Mittelwertes. (b) Steigender Fehler der Unsicherheit $\Delta\sigma$. (c) Änderung der Unsicherheit bei konstantem Fehler $\Delta\mu = 0.2$. (d) Änderung der Unsicherheit bei konstantem Fehler $\Delta\mu = 0.5$. In (c) und (d) ist zu erkennen, dass es im Falle eines fehlerhaften Mittelwertes keine konstante Änderung der Unsicherheit gibt, für die die Idealkurve erreicht wird.*

Im Falle der Kalibrierungskurve für die Regression liegt dieses Verhalten an der Annahme der Verteilung bei der Erstellung der Kurve. Für die hier erstellten Kalibrierungskurven wurde beispielsweise eine Normalverteilung der Fehler für eine Eingabe $x$ mit $\mathcal{N}(\mu(x), \sigma(x)$ angenommen. Für den Fall eines konstanten Fehlers des Mittelwertes $\mu$ würde sich die eigentliche Verteilung der Fehler verschieben, so dass diese durch eine Änderung der Unsicherheit $\sigma$ nicht mehr abgebildet werden kann. Die Abbildung 3.9 zeigt hierfür beispielhaft die Verschiebung der Verteilung der Fehler bei fehlerhafter Schätzung des Mittelwertes. Dargestellt ist hierfür die Verteilung der Fehler in einem speziellen Punkt $x$, wobei gilt $t_i \sim \mathcal{N}(\hat{\mu},1)$. $\hat{\mu}$ beschreibt dabei den eigentlichen Mittelwert der Zielwerte $t_i$. In dem dargestellten Fall kann die Normalverteilung $\mathcal{N}(0,\sigma)$ durch eine Unsicherheit $\sigma$ (hier: $\sigma = 1$) dargestellt werden, was zu einer idealen Kalibrierungskurve führt. Die rote Verteilung beschreibt die Verteilung der Fehler bei einem Fehler von $\Delta\mu = 6$. Hier kann die Verteilung nicht durch eine mittelwertfreie Normalverteilung beschrieben werden. Folglich entsteht ein Kalibrierungsfehler für alle möglichen Unsicherheiten $\sigma$.
Das zeigt zudem, dass die Unsicherheit bei der Kalibrierungskurve in erster Linie die Verteilung der Zielwerte $t_i$ abbilden soll. Diese Art der Unsicherheit wäre allerdings eine reine Datenunsicherheit, da die Verteilung der Zielwerte nicht vom Modell selbst anhängt. Eine alleinige Bewertung der Modellunsicherheit anhand der Kalibrierungskurve ist daher nicht sinnvoll, da diese die Unsicherheit verursacht durch die Daten nicht widerspiegelt.

### 3.4.2 Log-Likelihood

Eine oft verwendetes Qualitätsmaß zur Bewertung von Unsicherheiten ist die logarithmische Likelihood (LL) [Gal und Ghahramani, 2016b, Guo et al., 2017]. Diese trifft eine Aussage darüber, wie gut die geschätzten Parameter einer Verteilung an den tatsächlichen Parametern liegen. Für Regressionsprobleme wird dabei meist eine Likelihood basierend auf der Gauß-Verteilung verwendet. Das bedeutet, dass hier empirische Mittelwert und die empirische Standardabweichung bewertet wird. Die Formel für die Likelihood eines bestimmten Datenpunktes $(t,x)$ ergibt sich in diesem Fall zu:

$$L(t \mid x, \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{||t-y||^2}{2\sigma^2}} \tag{3.18}$$

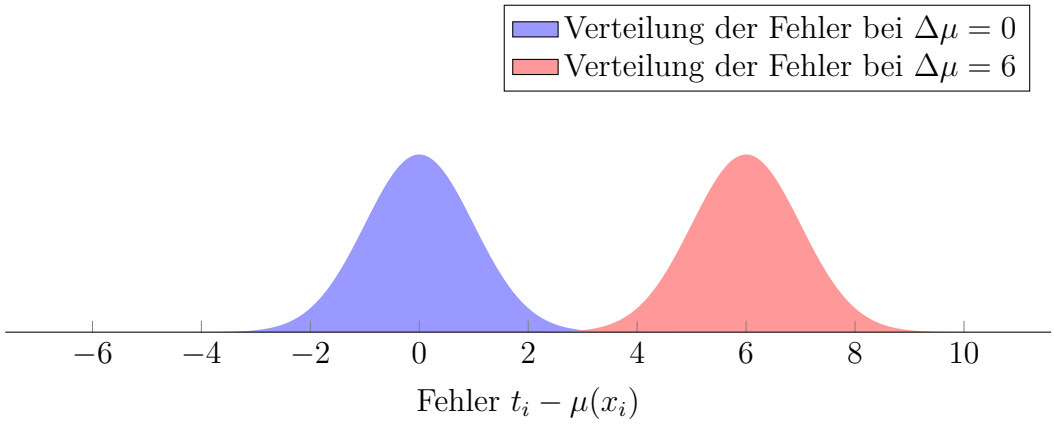



**Abbildung 3.9:** Beispielhafte Darstellung der Verschiebung der Verteilung der Fehler bei fehlerhafter Schätzung des Mittelwertes.
*Zu sehen ist die Darstellung Verschiebung der Fehlerverteilung bei fehlerhafter Mittelwertschätzung. Die blaue Verteilung zeigt dabei die Verteilung bei perfekter Schätzung des Mittelwertes.*

Wobei $\theta$ die Modellparameter und $(y,\sigma)$ die unter den Parametern $\theta$ berechnete Ausgabe des, bestehend aus der Prädiktion des Mittelwertes $y$ und der Unsicherheit $\sigma$ Modells darstellen.
Für die Klassifikation wird die über die Softmax-Ausgabe gegebene diskrete Verteilung verwendet. Die Likelihood ist hier definiert durch:

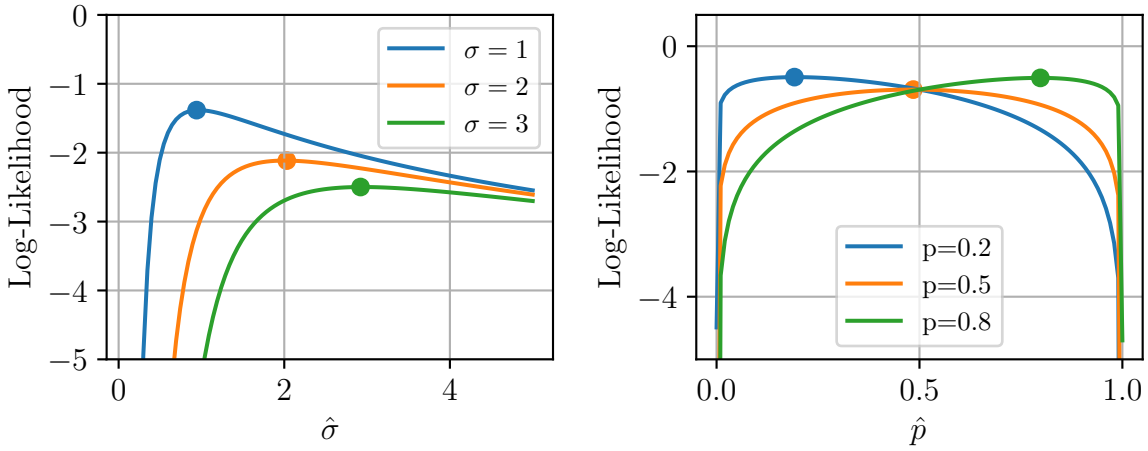
$$L(c \mid x,\theta) = f_{\theta,c}(x) \tag{3.19}$$

Wobei $f_{\theta,c}(x)$ die berechnete Wahrscheinlichkeit der Klasse $c$ für die Eingabe $x$ unter den Modellparametern $\theta$ darstellt. Die bei der Klassifikation verwendete Fehlerfunktion auf Basis der Kreuzentropie entspricht also der negativen Log-Likelihood.
Bei der Berechnung der Likelihood über mehrere unbekannte Datenpunkte kann dabei die zu erwartende Likelihood eines unbekannten Datenpunktes $(x^*, t^*)$ berechnet werden, indem für jeden Punkt in dem Datensatz die Likelihood berechnet wird und anschließend der Mittelwert gebildet wird. So ergibt sich folgende Formel:

$$L(t^*|x^*, \theta) = \frac{1}{N} \sum_{i=1}^{N} L(t_i^*|x_i^*, \theta) \tag{3.20}$$

wobei $(x_i^*, t_i^*)$ ein unbekannter Datenpunkt aus dem Datensatz $\{(x_i^*, t_i^*)\}_{i=1}^N$ ist, $\theta$ die Modellparameter und $L$ die Likelihood-Funktion darstellen. Der resultierende Wert beschreibt nun die zu erwartende Likelihood des Modells für einen unbekannten Datenpunkt.

**Verlauf der Log-Likelihood-Funktionen** Für die Bewertung eines Modells anhand der Log-Likelihood ist es wichtig zu wissen, wie der Verlauf dieser aussieht. Die Abbildung 3.10a zeigt einen Verlauf der Log-Likelihood-Funktion basierend auf der Normalverteilung in Abhängigkeit der geschätzten Standardabweichung. Die Daten, auf denen diese Abbildung erstellt wurde, entspringen jeweils einer Normalverteilung mit $\sigma \in [1,2,3]$. Für die Log-Likelihood wird der Mittelwert als konstant $\mu = 0$ angenommen und lediglich die Standardabweichung im Bereich $\hat{\sigma} \in [0.1, 5]$ variiert. Wie zu erwarten erhält man den maximalen Wert für die Log-Likelihood durch $\hat{\sigma} = \sigma$, wie an den durch die Punkte markierten Maxima zu erkennen ist. Zu beachten ist allerdings das Verhalten links bzw. rechts des Maximums. Für sehr kleine Werte für




**(a)** Regression (Gauß) **(b)** Klassifikation (Bernoulli)

**Abbildung 3.10:** Verlauf der Log-Likelihood-Funktionen für Regression und Klassifikation

*Zu sehen ist jeweils der Verlauf für unterschiedliche Daten mit unterschiedlichen Rauschtermen ($\sigma \in [1,2,3]$ für die Gauß-basierte Log-Likelihood und $p \in [0.3,0.5,0.8]$ für die Bernoulli-basierte Log-Likelihood. Das Maximum ist jeweils durch einen Punkt markiert.)*

$\hat{\sigma}$ fällt die Log-Likelihood stark ab, wohingegen für größer werdende Werte für $\hat{\sigma}$ die Log-Likelihood nur sehr langsam abfällt. Das hat zur Folge, dass Modelle, die ihre Unsicherheit stark überschätzen aber den Mittelwert gut approximieren anhand der Log-Likelihood schwerer vom Optimum zu unterscheiden sind, als Modelle, die ihre Unsicherheit unterschätzen. Anhand der Abbildung ist zu erkennen, dass je größer das tatsächliche $\sigma$ ist, desto flacher wird der Verlauf der Log-Likelihood. Dadurch verstärkt sich der oben beschriebene Effekt.
Die Abbildung 3.10b zeigt den Verlauf der in der Klassifizierung verwendeten Log-Likelihood basierend auf der Bernoulli-Verteilung. Die Datensätze beschreiben jeweils ein binäres Klassifikationsproblem mit Wahrscheinlichkeit für Klasse 1 $p \in 0.2{,}0.5{,}0.8$. Die Vorhersage für jeden Punkt im Datensatz ist dabei konstant $\hat{p}$. Erkennbar ist, dass die Log-Likelihood an den Grenzen des Wertebereichs für $\hat{p}$ absinkt. Für den Fall der maximal verrauschten Daten $p = 0.5$ ist die Log-Likelihood symmetrisch um das Maximum.

## 3.5 Zusammenfassung

Die Thematik der Unsicherheit bei der Parameterschätzung ist weit und geht weit zurück. In diesem Kapitel wurden die verschiedenen Ansätze dabei in drei Gruppen aufgeteilt, die bayesschen, approximierenden bayesschen und die nicht-bayesschen Verfahren. Die beiden ersten Gruppen basieren dabei auf der bayesschen Statistik. Die Verfahren aus der Gruppe der bayesschen Ansätze sind dabei in der Lage Stichproben der exakten *A-posteriori-Verteilung* zu bilden. Allerdings verhindert deren Ressourcenaufwand eine Anwendung auf realen Problemstellungen, da dort meist große Datensätze und Modellarchitekturen Verwendung finden. Die approximierenden bayesschen Verfahren, versuchen im Gegensatz dazu nicht mit der exakten *A-posteriori-Verteilung* zu arbeiten, sondern mit einer Approximation dieser. Dies ermöglicht eine Skalierung der Verfahren für komplexere Anwendungen auf Kosten der Genauigkeit der geschätzten Unsicherheiten. Die nicht-bayesschen Verfahren konzentrieren sich stark auf die Skalierbarkeit und versuchen, beispielsweise durch Ensembles, eine Repräsentation der Unsicherheiten zu erzeugen. Besonders in den beiden letzten Themengebieten wird derzeit viel geforscht, was zur Folge hat, dass die aktuellsten Arbeiten in der State-of-the-Art-Recherche aus diesen Gebieten stammen.

Für diese Masterarbeit können allerdings nicht alle Verfahren erprobt werden, daher muss eine Auswahl für die späteren Experimente erfolgen. Dabei scheiden bayessche Ansätze vorerst aus, da möglichst Verfahren gewählt werden sollen, welche leicht in der Praxis anzuwenden sind. Daher wird das Verfahren MC-Dropout nach [Gal und Ghahramani, 2016b] für eine Schätzung der Modellunsicherheit und das Verfahren nach [Kendall und Gal, 2017] für eine Schätzung der Datenunsicherheit ausgewählt. Damit stehen Methoden zur Schätzung beider Unsicherheiten zur Verfügung.
Ein großer Vorteil beider Verfahren ist dabei der geringe Aufwand bei der Implementierung und deren Skalierbarkeit auf größere Datensätze oder Modellarchitekturen. Allein für MC-Dropout fällt in der Anwendung ein erhöhter Ressourcenaufwand an, da ein Modell mehrmals berechnet werden muss. Zudem können beide Verfahren gleichzeitig verwendet werden, um die verschiedenen Unsicherheiten auch in der Praxis trennen zu können.
Um die berechneten Unsicherheiten bewerten zu können wurden zudem in Abschnitt 3.4 (Seite 32) zwei Werkzeuge vorgestellt, welche im State of the Art Verwendung finden, um eine qualitative Aussage über Unsicherheiten machen zu können. Doch eine Bewertung der Modellunsicherheit anhand dieser Werkzeuge stellt sich als unzureichend heraus. Für beide Metriken gilt, dass diese bewerten, wie gut die Unsicherheiten die Fehler des Modells erklären. Allerdings muss die Modellunsicherheit diese Eigenschaft nicht erfüllen. Die Tatsache, dass die Modellunsicherheit abhängig von der Anzahl der Datendichte ist, stellt einen Widerspruch dazu dar, dass diese Unsicherheiten proportional zu den Fehlern des Modells sein sollen[15].
Aus diesem Grund wird in dieser Arbeit zusätzlich ein Verfahren aus der Gruppe der bayesschen Ansätze verwendet, um eine möglichst verlässliche Schätzung der Modellunsicherheit zu erhalten. Bei diesem Verfahren handelt es sich um den NUTS [Hoffman und Gelman, 2014], da dieser eine automatische Hyperparameterschätzung realisiert, was dessen Anwendung erleichtert. Die von diesem Verfahren geschätzten Modellunsicherheiten können als Vergleich für die Bewertung von MC-Dropout herangezogen werden.
Neben diesem Vergleich können die Unsicherheiten zudem in verschiedenen Anwendungen getestet werden, um deren praktischen Nutzen zu erproben. In dieser Arbeit werden die Unsicherheiten deshalb für die Anomaliedetektion verwendet.

[15] Eine detailliertere Erläuterung dazu ist im Anhang A.1.3 zu finden.

# Kapitel 4

# Verwendete Datensätze

In diesem Kapitel werden die verwendeten Datensätze näher beschrieben, auf denen die zuvor ausgewählten Verfahren zur Schätzung von Unsicherheiten getestet wurden. Zu Beginn wird auf einem selbst erstellten Datensatz gearbeitet. Vorteil hierbei ist, dass Informationen, die zur Bewertung von Schätzungen von Unsicherheiten benötigt werden, bekannt sind. Die Parameter der verwendeten Rauschterme bilden hier ein wichtiges Beispiel. Dadurch lassen sich eindeutige Aussagen über die Qualität der Schätzungen machen, was bei einem realen Datensatz nur schwer möglich ist.
Durch die Erkenntnisse, die auf den synthetischen Daten gewonnen werden können, kann danach eine Evaluierung auf einem realen Datensatz stattfinden. Die jeweiligen Zielstellungen unterscheiden sich dabei für die beiden Arten von Daten. Die Verwendung der synthetischen Daten erfüllt in erster Linie den Zweck, ein tieferes Verständnis über die verschiedenen Arten von Unsicherheiten zu gewinnen und erste Aussagen über die gewählten Verfahren zu machen. Für den Realdatensatz, der aus Prozessdaten der Firma STEAG besteht, liegt das Ziel darin, zu untersuchen, ob Anomalien in den Messwerten anhand der Unsicherheiten entdeckt werden können. Hierfür werden zwei verschiedene Herangehensweisen erläutert und die dabei verwendeten Datensätze beschrieben.

## 4.1 Synthetische Daten

Da für den Eingabebereich sowohl negative als auch positive Werte gewünscht waren, um Abhängigkeiten vom Vorzeichen der Eingabe ausschließen zu können, bietet sich ein

generierende Funktion an, welche symmetrisch im Punkt $x = 0$ ist. Daher wurde sich für die Kosinusfunktion entschieden, um die Zielwerte des synthetischen Datensatzes zu generieren. Die periodische Eigenschaft der Kosinusfunktion kann zudem verwendet werden, um Auswirkungen in der Änderung der Daten zu untersuchen, während sich die grundlegende Funktion nicht ändert. Zum Beispiel kann die Auswirkung von verschiedenen Rauschtermen untersucht werden, indem verschiedene Bereiche der Kosinusfunktion auf unterschiedliche Art verrauscht werden. Die formale Beschreibung der generierenden Funktion kann also folgendermaßen beschrieben werden:

$$y = \cos(x) + \epsilon(x) \tag{4.1}$$

wobei $\epsilon(x)$ den Rauschterm darstellt, welcher im Folgenden genauer definiert wird. Für den Wertebereich von $x$ wurde sich zudem auf das Intervall $[-4\pi, 4\pi]$ beschränkt.
Um einen Datensatz zu generieren, anhand dessen die Qualität der geschätzten Unsicherheiten bewertet werden kann, wurden die benötigten Eigenschaften der Daten anhand der Definitionen der Unsicherheiten in Abschnitt 2.1 (Seite 5) abgeleitet.
Folgend werden die besonderen Eigenschaften in Bezug auf die unterschiedlichen Unsicherheitstypen (Modellunsicherheit und Datenunsicherheit) näher beschrieben.

**Modellunsicherheit** Für die Modellunsicherheit gilt laut Definition, dass diese mit der Zahl der Datenpunkte abnimmt. Somit ergeben sich zwei Situationen für die eine Schätzung der Modellunsicherheit bewertet werden kann. Zum einen kann durch eine unterschiedliche Datendichte bewertet werden, ob ein Verfahren mit steigender Datenanzahl eine kleinere Modellunsicherheit schätzt. Zum anderen kann das Verhalten der Modellunsicherheiten an den Rändern des Bereiches, aus dem die Trainingsdaten stammen, untersucht werden. Dabei sollte die Modellunsicherheit für unbekannte Daten größer sein als für Daten, die in der Nähe der Trainingsdaten liegen. Somit wurde die Verteilung der Trainingsdaten nicht als Gleichverteilung gewählt, sondern als linear ansteigende Funktion. Die Punkte am linken Rand des Intervalls $[-4\pi, 4\pi]$ besitzen dabei die geringste Auftrittswahrscheinlichkeit und die Punkte am rechten Rand die größte. Um die Modellunsicherheit in Abhängigkeit der Datendichte vergleichen zu können wurden Bereiche definiert, welche in Tabelle 4.1 beschrieben sind. Dabei beschreiben vier dieser Bereiche unbekannte Daten und zwei beinhalten die bekannten Daten auf

| Intervall | Bezeichner | Beschreibung |
|---|---|---|
| $[-6\pi, -5\pi]$ | G_weit_außen | Unbekannt, weit entfernt |
| $[-5\pi, -4\pi]$ | G_außen | Unbekannt, direkt an bekannten Daten mit geringer Dichte angrenzend |
| $[-4\pi, 0]$ | G_innen | Bekannt, geringe Datendichte |
| $[0, 4\pi]$ | H_innen | Bekannt, höhere Datendichte |
| $[4\pi, 5\pi]$ | H_außen | Unbekannt, direkt an bekannten Daten mit hoher Dichte angrenzend |
| $[5\pi, 6\pi]$ | H_weit_außen | Unbekannt, weit entfernt |

**Tabelle 4.1:** Beschreibung der Eingabebereiche.
*In dieser Tabelle sind die einzelnen Bereiche definiert. Dafür wird für jeden Bereich das Intervall, der verwendete Bezeichner und eine Kurzbeschreibung angegeben. Die Bezeichner werden hauptsächlich im Abschnitt 5.1.3 (Seite 72) verwendet.*

denen die Modelle trainiert werden. Zur Veranschaulichung der Verteilung und der Datenbereiche sind in Abbildung 4.1 Histogramme für die Trainings-, Validierungs- und Testdaten dargestellt. Zu erkennen ist die ansteigende Verteilungsfunktion für die Trainings- bzw. Validierungsdaten. Das Histogramm der Testdaten im Bereich der bekannten Daten zeigt keine glatte Gleichverteilung, da sichergestellt werden musste, dass sich die Punkte aus dem Trainings- und Testset nicht zu stark ähneln. Dabei wurden zuerst 25000 Testdaten im Trainingsbereich mit gleicher Schrittweite (0.0015) erzeugt. Danach wurden alle Datenpunkte entfernt, welche einen Mindestabstand von 0.0001 zu einem Datenpunkt aus dem Trainings- bzw. Validierungsset unterschritten. Eine genauere Untersuchung der Auswirkungen der verwendeten Schrittweite für die Testdaten oder des Mindestabstands der Datenpunkte konnte im Rahmen dieser Arbeit nicht untersucht werden.
Die sich ergebenden Datensatzgrößen sind in Tabelle4.2 aufgelistet.

**Datenunsicherheit** Für die Definition des Rauschterms $\epsilon(x)$ in der Formel 4.1 wurden zwei verschiedene Varianten verwendet. Zum einen wurde ein varianzhomogener Rauschterm $\epsilon(x) \sim \mathcal{N}(0{,}0.1)$ verwendet. Begründet ist dies durch die möglichen Auswirkungen, die ein varianzheterogener Rauschterm auf die geschätzte Modellunsicherheit

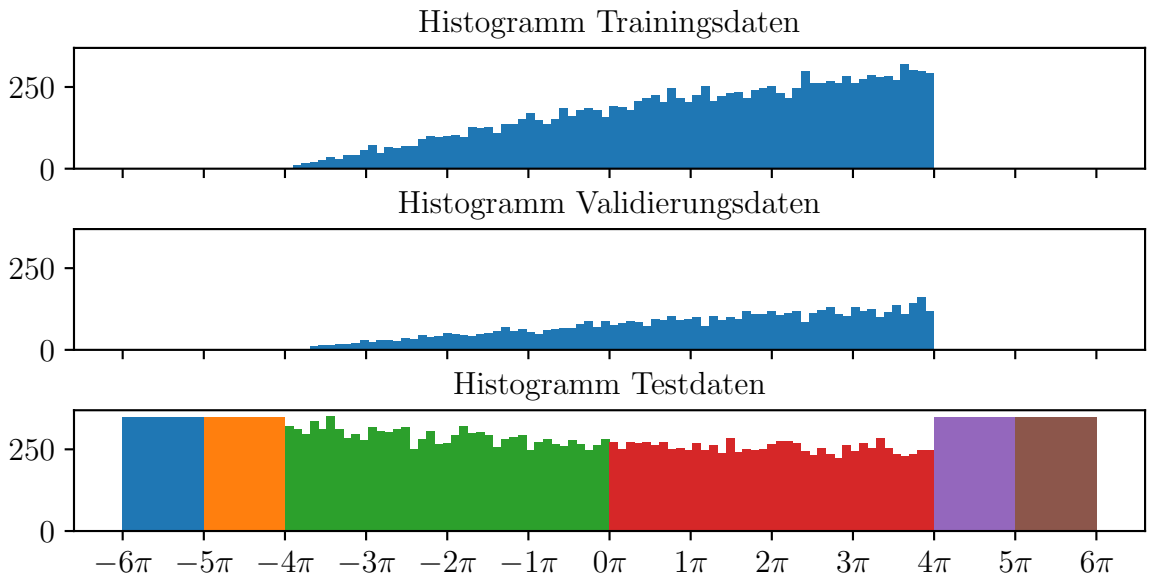
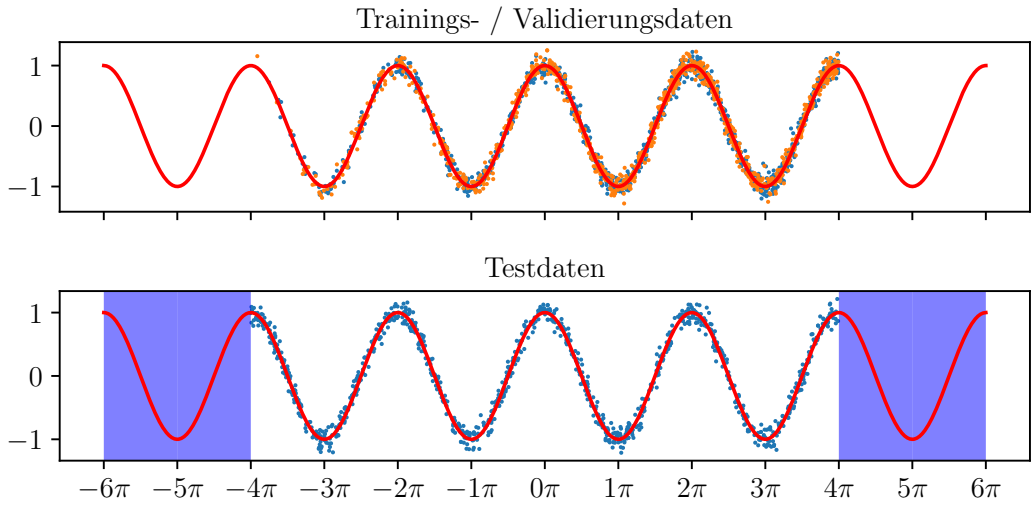



**Abbildung 4.1:** Histogramm der Trainings-, Validierungs- und Testdaten für den varianzhomogenen Datensatz.
*Die Abbildung zeigt die Histogramme der Trainings-, Validierungs- und Testdaten der synthetischen Daten. Der Testdatensatz ist dabei unterteilt in die zuvor definierten Bereiche, welche im Histogramm farblich gekennzeichnet sind.*

haben könnte. Durch einen varianzhomogenen Datensatz kann sich bei diesem auf die Bewertung der Modellunsicherheit fokussiert werden. Um die Verfahren zur Schätzung von varianzheterogener Datenunsicherheit evaluieren zu können wurde ein zweiter Datensatz mit Rauschterm verwendet, bei dem die Standardabweichung für jede halbe Periode der Kosinusfunktion zwischen den Werten 0.5 und 0.1 alterniert wird. Die beiden Datensätze sind in der Abbildung 4.2 dargestellt.

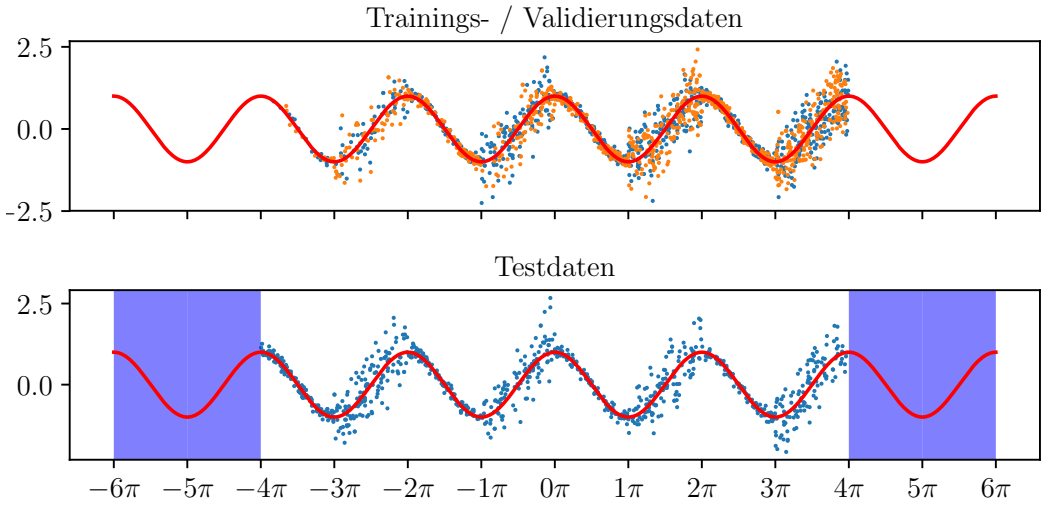
| Datensatz | Größe |
|---|---|
| Trainingsdaten | 13405 |
| Validierungsdaten | 5745 |
| Testdaten | 21199 |

**Tabelle 4.2:** Angaben zur Datensatzgröße




**(a)** Varianzhomogener Datensatz.

*Datensatz mit konstantem Rauschterm* $\mathcal{N}(0, 0.1)$.




**(b)** Varianzheterogener Datensatz.

*Datensatz mit alternierendem Rauschterm* $\mathcal{N}(0, 0.1)$ *bzw* $\mathcal{N}(0, 0.5)$.

**Abbildung 4.2:** Streudiagramme der beiden synthetischen Datensätze.

*Die Abbildung zeigt Scatter-Plots für beide Datensätze (Oben: Varianzhomogener Datensatz, Unten: Varianzheterogener Datensatz). Die jeweils obere Abbildung zeigt die Trainings- (blau) und Validierungsdaten (orange). Die untere zeigt die Testdaten auf denen die Modelle letztendlich ausgewertet werden. Die blauen Bereiche repräsentieren dabei Bereiche aus denen unbekannte Daten gezogen wurden. Die rote Linie zeigt die zugrundeliegende Kosinusfunktion.*

## 4.2 Prozessdaten der STEAG GmbH

Die Prozessdaten der Firma STEAG GmbH werden in dieser Arbeit als Realbeispiel eines Regressionsproblems verwendet. Diese Prozessdaten beschreiben dabei die gemessenen Werte der verschiedenen Prozessgrößen eines industriellen Prozesses. Da in dieser Arbeit unabhängig von der Art der Messgrößen gearbeitet werden soll, wird die Bedeutung der einzelnen Messgrößen vernachlässigt. Die Messkanäle, welche als Modellausgaben verwendet werden, werden dabei durchnummeriert. Der Datensatz besteht aus den Aufzeichnungen über einen Zeitraum von fünf Jahren. Durch die Eigenschaften des zugrunde liegenden Prozesses zeigt sich ein periodisches Verhalten über die Jahre hinweg, da dieser zum Beispiel durch die Umgebungstemperatur beeinflusst wird. Für die weitere Verwendung des Datensatzes kann diese Eigenschaft allerdings vernachlässigt werden, da sich dies bei vorherigen Untersuchungen der Firma STEAG als unerheblich gezeigt hat. Jeder Datenpunkt beschreibt einen Mittelwert der Messgrößen, welcher über ein Zeitfenster von einer Stunde gebildet wurde. Dieses Vorgehen soll Rauschen, welches zum Beispiel durch die verwendeten Messgeräte verursacht wird, beseitigen. Somit ergeben sich über fünf Jahre 42140 Datenpunkte für den Datensatz. Da für diese Daten eine Untersuchung analog zu der auf den synthetischen Daten nicht möglich ist, da Informationen wie beispielsweise die exakten Rauschterme für die Datenunsicherheit nicht vorhanden sind, wird hier eine andere Problemstellung formuliert. Auf diesen Daten soll anhand der Unsicherheiten eine Detektion von Anomalien untersucht werden. Hierfür stehen für elf der insgesamt 233 Messgrößen zusätzlich Informationen in binärer Form zur Verfügung, welche beschreiben, ob ein Datenpunkt als Anomalie gilt oder nicht. Diese elf Messgrößen werden deshalb als Ausgabekanäle für die späteren Modelle verwendet. Diese Labelinformationen wurden zuvor durch Methodiken der *Stochastische Prozesslenkung* (englisch: stochastic process control (SPC)) bestimmt. Grundsätzliches Vorgehen hierbei ist wie folgt definiert:

- Ein Modell (z.B. ein Neuronales Netz) wird auf einem Trainingsdatensatz trainiert.
- Die Ausgabe dieses Modells wird für einen neuen Datensatz bestimmt, für den Anomalien detektiert werden sollen.
- Anhand eines Vergleichs der Modellausgabe und den gemessenen Werten wird nun anhand einer Heuristik bestimmt, ob ein Punkt als Anomalie gilt.

Der Trainingsdatensatz muss zur Detektion der Anomalien aus normalen Daten beste-

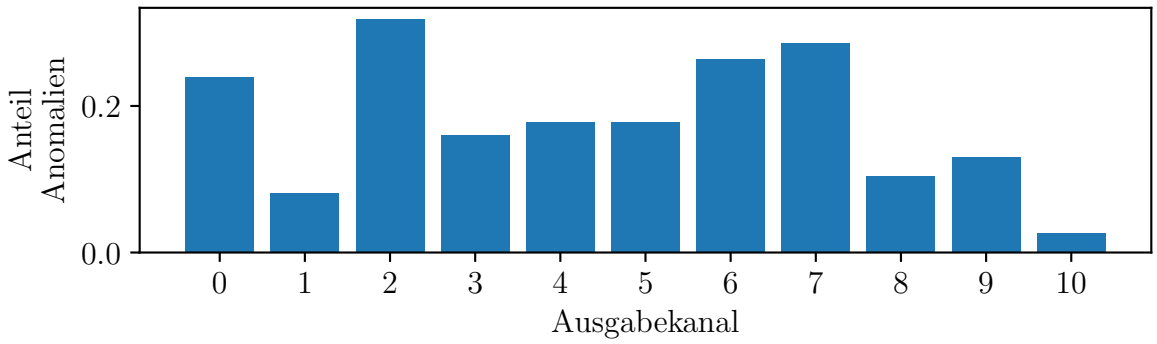



**Abbildung 4.3:** Anteil der Anomalien pro Ausgabekanal.
*Die Abbildung zeigt den Anteil der Anomalien für jeden der verfügbaren Ausgabekanäle als Balkendiagramm.*

hen. Das bedeutet, dass im Vorfeld Informationen über den Anomaliestatus der Daten vorhanden sein muss.
Anhand einer Heuristik über dem Fehler des trainierten Modelles wird nun entschieden, ob ein unbekannter Datenpunkt als Anomalie gekennzeichnet wird. Beispielsweise muss der Fehler des Modells über ein bestimmtes Zeitfenster groß genug sein, damit weitere Punkte mit hohem Fehler als Anomalie gelten. Diese zur Detektion verwendete Heuristik folgt dabei dem Verfahren sequential probability ratio test (SPRT) [Wald, 1945]. Da dieses jedoch nicht Thema dieser Masterarbeit ist, wird dieses nicht näher erläutert.
Der Anteil der Anomalien pro Ausgabekanal ist in Abbildung 4.3 zu sehen. Zusätzlich ist anzumerken, dass der Anteil der Daten bei denen mindestens einer der Ausgaben als Anomalie gekennzeichnet ist 85.6% ausmacht. Da sich eine Anomalie in den betrachteten industriellen Prozessen grundsätzlich nicht nur auf eine Messgröße beschränkt kann daher angenommen werden, dass der Großteil des Datensatzes aus Datenpunkten besteht, welche eine Anomalie im Gesamtsystem beschreiben. Diese Tatsache ist besonders für den Ansatz der Anomaliedetektion anhand der Modellunsicherheit, welcher im Laufe dieses Kapitels näher beschrieben wird, von Bedeutung.
Da die elf Messgrößen mit Anomalie-Informationen als Modellausgabe verwendet werden, bleiben die restlichen 222 Messgrößen als Eingabekanäle übrig. Dabei werden immer alle Eingabekanäle verwendet, um keine aufwendige Auswahl der Modelleingaben vornehmen zu müssen. Dies gilt besonders für Modelle, welche nur auf einem

der elf Ausgabekanäle trainiert werden. Das bedeutet allerdings, dass es Eingaben gibt, welche keinen wirklichen Zusammenhang zu den Ausgabekanälen haben. Diese unwichtigen Eingaben müssen von den Modellen eigenständig erkannt werden. Dennoch ist dieses Vorgehen gewünscht, da so keine Kenntnis des unterliegenden Prozesses und der Zusammenhänge zwischen den Prozessgrößen vorausgesetzt wird.

### 4.2.1 Vorverarbeitung

Bevor ein Modell auf diesen Daten trainiert werden kann, müssen diese zuerst für diese Modelle vorbereitet werden. Zum einen wurden die einzelnen Messgrößen standardisiert, das bedeutet, dass jede Messgröße $x$ anhand folgender Formel transformiert wurde:

$$x \leftarrow \frac{x - \hat{\mu}}{\hat{\sigma}} \tag{4.2}$$

Die dafür notwendigen Werte für $\hat{\mu}$ und $\hat{\sigma}$ wurden anhand der Trainingsdaten bestimmt und für alle Teildatensätze verwendet. Durch die unterschiedlichen Verteilungen der verschiedenen Messgrößen werden zudem die Varianzen der Ausgabekanäle bei der Berechnung der Fehlerwerte der Modelle berücksichtigt. Das bedeutet, dass der Fehler eines Modelles über einem Kanal mit Zielwerten $T = \{t_i\}_i^N$ durch folgende Formel berechnet wird:

$$Err = \frac{1}{N} \sum_{i=1}^{N} \frac{(t_i - y_i)^2}{\mathrm{Var}(T)} \tag{4.3}$$

Dabei beschreibt $y_i$ die Ausgabe des Modells für den $i$-ten Datenpunkt. Dieser Fehler entspricht dem durch die Varianz skaliertem Mean Squared Error (MSE) und wird deshalb in den Experimenten als „MSE“ bezeichnet.
Zum anderen muss bei der Aufteilung der Daten in Trainings-, Validierungs- und Testset darauf geachtet werden, dass sich Trainingsset und Testset ausreichend unterscheiden, da zwei zeitlich aufeinander folgende Punkte in der Regel recht ähnlich sind. Um dies in gewissem Maße zu gewährleisten wurde der Datensatz in Wochenabschnitte unterteilt. Das bedeutet der Datensatz wurden in 251 Blöcke mit jeweils 168 Datenpunkte (7 Tage$\cdot$ 24 Stunden) eingeteilt. Diese Blöcke wurden nun zufällig zu Trainings-, Validierungs- oder Testset zugeordnet. Dabei wurde ein Verhältnis von 60% Trainingsdaten, 10% Validierungsdaten und 30% Testdaten angestrebt. Da sich die Datensätze für die

verschiedenen Vorgehensweisen unterscheidet werden die genauen Details erst im folgenden Abschnitt aufgelistet.

### 4.2.2 Vorgehen bei der Anomaliedetektion

Da in dieser Arbeit zwei Arten von Unsicherheiten unterschieden werden, muss die Vorgehensweise bei der Detektion von Anomalien für jede der beiden Unsicherheiten angepasst werden. Neben den Anpassungen kann der allgemeine Ablauf der Experimente allerdings wie folgt beschrieben werden:

- Training eines Modells auf den Messdaten als Prognosemodell.
- Bestimmung der Unsicherheit auf dem Testdatensatz.
- Die bestimmte Unsicherheit kann nun als Score für die Klassifikation der Anomalien verwendet werden (Die Bewertung des resultierenden Klassifikators wird anhand von ROC-Kurven vorgenommen um sich nicht auf einen festen Schwellwert für die Unsicherheiten festlegen zu müssen).

Wichtig hierbei ist, dass das Modell als Prognosemodell trainiert wird, das bedeutet, dass die Ausgabe des Modells eine Schätzung des nächsten Messwertes aller Ausgaben ist und keine direkte Klassifikation.
Im Folgenden werden nun die unterschiedlichen Vorgehensweisen für die beiden Typen der Unsicherheit genauer beschrieben. Dabei werden die Annahmen definiert, auf denen die beiden Ansätze arbeiten und etwaige Anpassungen bei dem zuvor beschriebenen Ablauf der Untersuchungen beschrieben.

**Detektion anhand der Datenunsicherheit** Für eine Detektion der Anomalien anhand der Datenunsicherheit gilt die Grundannahme, dass Anomalien einen Widerspruch zu den gewöhnlichen Zusammenhängen im Datensatz bilden. Dieser Widerspruch äußert sich in unterschiedlichen Zielwerten für gleiche Eingaben. Zur Veranschaulichung zeigt Abbildung 4.4 normale Datenpunkte und Anomalien eines hypothetischen Systems mit Eingabe $x$ und Ausgabe $y$. Dabei stellen die orangen Anomalien in einem bestimmten Bereich des Eingaberaums einen Widerspruch zu den blauen normalen Daten dar. Anzumerken ist, dass für eine simplere Veranschaulichung angenommen ist, dass sich Anomalien in einem bestimmten Bereich der Eingabe konzentrieren. Für ein Modell, welches die Datenunsicherheit explizit modelliert, sollte demnach die Datenunsicherheit

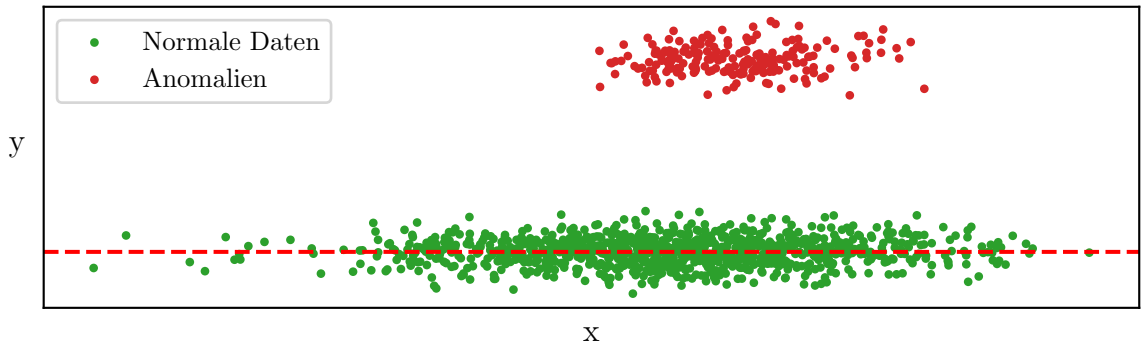



**Abbildung 4.4:** Visualisierung der Annahme an die Anomalien für die Detektion mit Datenunsicherheit
*Zu sehen ist eine Visualisierung der Grundannahme an die Anomalien, welche anhand der Datenunsicherheit erkannte werden sollen. Die rote Linie stellt den Zusammenhang zwischen Eingabe $x$ und Ausgabe $y$ dar. Die grünen Punkte stellen dabei normale Messungen dar, während die roten Datenpunkte Anomalien darstellen.*

für Anomalien steigen, da es dieser Widerspruch unmöglich macht eine exakte Lösung zu geben. Allerdings würden die normalen Daten, die Teil des Widerspruchs sind (im Bild direkt unter den Anomalien), auch eine dementsprechend hohe Datenunsicherheit zugeteilt bekommen und deshalb fälschlicherweise als Anomalien klassifiziert. Das bedeutet, dass für diesen Ansatz eine hohe Falsch-Positiv-Rate erwartet wird.
Da dem Modell die Widersprüche während des Trainings zur Verfügung stehen müssen, muss der Trainingsdatensatz sowohl aus Anomalien und normalen Datenpunkten bestehen. Diese Tatsache hat den Vorteil, dass für den Anwendungsfall im Vorfeld keine Kenntnis über die Anomalien nötig ist. Ein Modell kann somit auf einem beliebigen Datensatz trainiert werden, solange ausgeschlossen werden kann, dass sich in dem Datensatz keine Anomalien befinden, wenn diese anhand der Datenunsicherheit erkannt werden sollen. Wie die Angaben zu den vorhandenen Daten im Abschnitt 4.2 allerdings zeigen, ist der Großteil der Daten als Anomalie gekennzeichnet und somit sollte dies kein Problem sein.
Grundsätzlich ist dieser Ansatz ähnlich zu dem von der STEAG Energy Services GmbH (SES) verwendeten Vorgehen. In beiden Fällen werden Punkte als Anomalie bezeichnet, wenn das verwendete Modell einen größeren Fehler auf diesen macht. Dies spricht für positive Erwartungen an die Anomaliedetektion anhand der Datenunsicherheit, allerdings bedeutet das auch, dass die Ergebnisse in dieser Arbeit abhängig vom

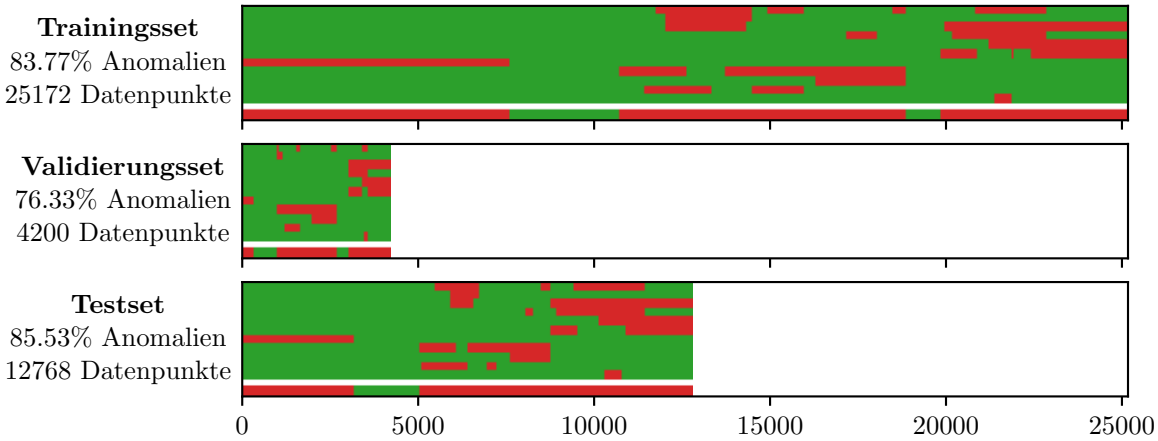



**Abbildung 4.5:** Visualisierung der Aufteilung in normale Daten und Anomalien. *Für jeden der Teildatensätze ist hier die Aufteilung in normale Daten und Anomalien visualisiert. Für diese Visualisierung wurden die Datenpunkte zeitlich sortiert. Entlang der x-Achse ist jeder Datenpunkt in rot, falls dieser eine Anomalie darstellt, oder in grün, falls es sich um einen normalen Datenpunkt handelt, abgetragen. Die letzte separate Zeile markiert jeweils die Datenpunkte in rot, für die mindestens einer der Ausgabekanäle als Anomalie gekennzeichnet ist. Auf diese Zeilen beziehen sich auch die Angaben des Anteils der Anomalien, welche in Prozent am Rande angegeben sind. Zusätzlich ist für jeden der Teildatensätze die Anzahl der Datenpunkte angegeben.*

Verfahren sind, welches zur Erzeugung der Labelinformationen verwendet wurde. Nach dem im Abschnitt 4.2.1 (Seite 52) beschriebenen Vorgehen können nun die Daten in die Teildatensätze eingeteilt werden. Da bei dieser Vorgehensweise der Datensatz nicht anhand der Anomalielabel getrennt wird, enthalten die verschiedenen Teildatensätze sowohl Anomalien als auch normale Daten. Die Abbildung 4.5 zeigt eine Visualisierung der Aufteilung der Daten. Wie zu erkennen ist, sind die Anteile der Anomalien in den einzelnen Teildatensätzen ähnlich groß.

**Detektion anhand der Modellunsicherheit** Bei der Detektion der Anomalien anhand der Modellunsicherheit gilt die Annahme, dass sich die Anomalien von den normalen Datenpunkten in der Eingabe unterscheiden. Abbildung 4.6 zeigt eine Visualisierung der getroffenen Annahme analog zu Abbildung 4.4. Das bedeutet, falls die normalen Daten als bekannt gelten, können diese von den Anomalien als unbekannte Daten anhand der Modellunsicherheit unterschieden werden. Dies stellt einen entschei-

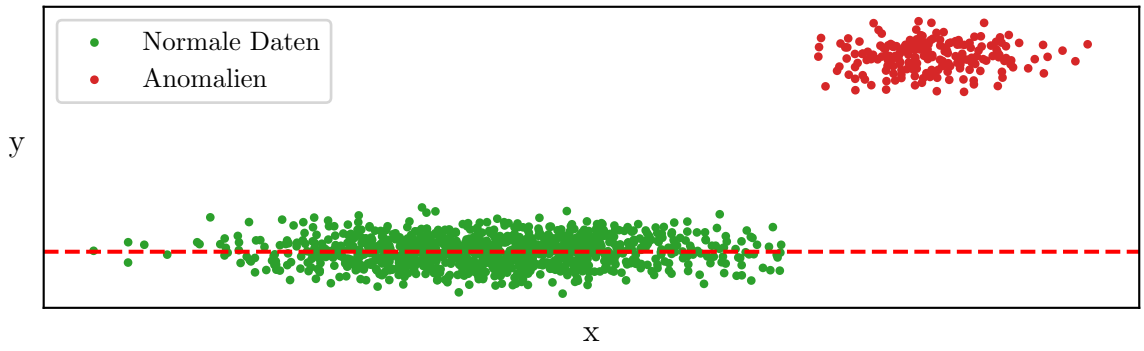



**Abbildung 4.6:** Visualisierung der Annahme an die Anomalien für die Detektion mit Datenunsicherheit
*Zu sehen ist eine Visualisierung der Grundannahme an die Anomalien, welche anhand der Datenunsicherheit erkannte werden sollen. Die rote Linie stellt den Zusammenhang zwischen Eingabe $x$ und Ausgabe $y$ dar. Die grünen Punkte stellen dabei normale Messungen dar, während die roten Datenpunkte Anomalien darstellen. Die Anomalien liegen dabei in einem anderen Bereich im Eingaberaum als die normalen Daten.*

denden Unterschied zu der Detektion anhand der Datenunsicherheit dar, da diese auf den Modellausgaben arbeitet.
Um Anomalien als unbekannt deklarieren zu können, darf das Modell allerdings nur auf normalen Daten trainiert werden. Das bedeutet, dass im Vorfeld Informationen über die Anomalien bekannt sein müssen, um einen Trainingsdatensatz erstellen zu können. Dies stellt einen entscheidenden Nachteil dieser Herangehensweise dar. Zudem kann sich in der Anwendung die Verteilung der Eingabekanäle auch bedingt durch die Umwelt (z.B. Änderung der Außentemperatur) ändern. In diesem Fall würde dies zu einer Klassifikation als Anomalie führen.
Vorteilhaft ist allerdings, dass sich dieses Vorgehen von der Methode unterscheidet, mit der die eigentlichen Labelinformationen erstellt wurden. Bei der Detektion von Anomalien anhand der Modellunsicherheit hat der Fehler des Modells auf die Klassifikationsentscheidung keinen Einfluss. Somit kann theoretisch eine andere Art der Anomalie erkannt werden als mit den bisherigen Verfahren, da ein Modell auf unbekannten Datenpunkten nicht zwangsweise einen großen Fehler macht.
Da für diese Herangehensweise allerdings ein Trainingsdatensatz bestehend aus ausschließlich normalen Daten notwendig ist ergeben sich folgende Optionen für den

Datensatz:

- Die Labelinformationen aller Ausgabekanäle wird mittels einer ODER-Verknüpfung verrechnet. Anhand dieses Ergebnisses werden die Daten in Anomalien und normale Daten aufgeteilt. Es wird nun für alle Ausgabekanäle ein einziges Modell trainiert.
- Es wird ein eigener Datensatz für jeden Ausgabekanal anhand dessen Labelinformationen erstellt. Für jeden Ausgabekanal wird daraufhin ein eigenes Modell trainiert.

Wie zuvor beschrieben wird eine Anomalie zwar nur anhand eines Ausgabekanals erkannt, jedoch bedeutet dies nicht, dass alle anderen Messwerte ein normales Verhalten aufweisen. Dies bedeutet, dass die Trainingsdatensätze für die zweite Option nicht zwangsweise anomaliefrei sind, da die Labelinformationen der anderen Ausgabekanäle ignoriert werden. Das bedeutet, dass die erste Variante zu bevorzugen ist. Zudem könnte die gleiche Modellarchitektur, wie bei der Anomaliedetektion anhand der Datenunsicherheit, genutzt werden.
Jedoch wurde sich in dieser Arbeit dennoch für die zweite Variante entschieden, da der Großteil der Daten als Anomalie gekennzeichnet sind. Wie bereits erwähnt machen die Anomalien 85.6% der Daten aus. Das bedeutet, dass ein Trainingsdatensatz aus normalen Daten sehr klein wäre[1]. Da der Anteil der Anomalien pro Ausgabekanal deutlich geringer ist, können für die zweite Option Datensätze erstellt werden, welche eine ausreichend große Anzahl an Daten besitzen.
Die Abbildung 4.7 zeigt die Aufteilung des Datensatzes in Trainings-, Validierungs- und Testdatensatz pro Ausgabekanal. Das Vorgehen bei der Einteilung der normalen Daten entspricht dabei dem Vorgehen aus Abschnitt 4.2.1. Die Anomalien wurden nachträglich dem Testset hinzugefügt, sodass sich im Trainings- und Validierungsset keine Anomalien befinden.

[1] Die Anzahl der Datenpunkte, für die keine der Ausgabekanäle als Anomalie gekennzeichnet ist beträgt 6928.

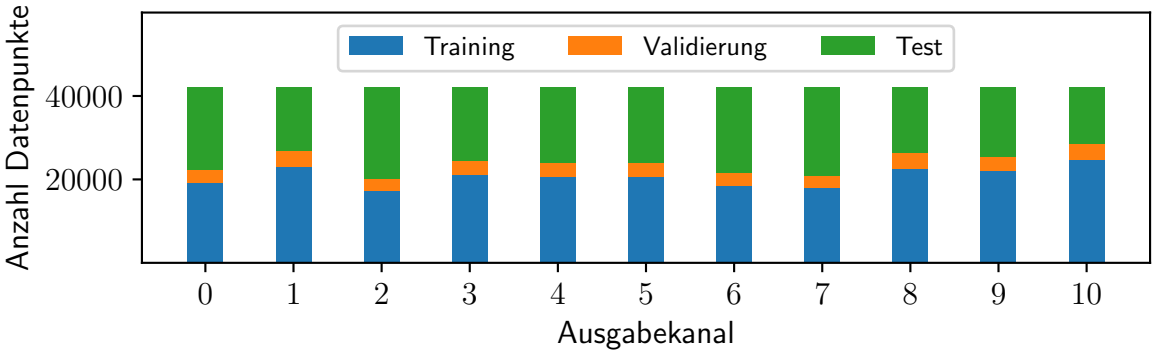



**Abbildung 4.7:** Aufteilung der Datensätze pro Ausgabekanal

*Dargestellt ist die Aufteilung der Datensätze für jeden Ausgabekanal in Trainings-, Validierungs- und Testdatensatz.*

# Kapitel 5

# Experimentelle Untersuchungen

In diesem Kapitel werden die Ergebnisse der einzelnen Verfahren auf den synthetischen Daten bzw. auf den Prozessdaten der STEAG Energy Services GmbH (SES) präsentiert. Da sich die Zielstellungen für den synthetischen Datensatz und den Realdaten unterscheiden, wird der Fokus der Experimente zuvor genau erläutert. Zudem wird jeweils das Vorgehen bei der Findung einer geeigneten Modellarchitektur beschrieben, da diese für beide Datensätze erst gefunden werden muss.
Um möglichst unabhängig von der Initialisierung eines Modells und weiteren stochastischen Einflüssen arbeiten zu können, wurde jedes Training fünf mal durchgeführt. Für die Experimente in diesem Kapitel wird deshalb jeweils die Verteilung über diesen Durchläufen präsentiert. Dabei sind jeweils fünf p-Quantile mit $p \in \{0, 0.25, 0.5, 0.75, 1\}$ abgetragen[1].
Für die Darstellung von Kurven wird das 0.5-Quantil (Median) dabei als durchgezogene Linie repräsentiert, während für die restlichen Quantile durch Flächen verschiedener Helligkeitsstufen markiert sind. Das Ende der dunklen Fläche beschreibt dabei das 0.25- bzw. 0.75-Quantil und das Ende der helleren Fläche beschreibt das 0- bzw. 1-Quantil. Bei präsentierten Balkendiagrammen wird die Höhe eines Balkens durch den Median bestimmt, während die restlichen Quantile durch Fehlerbalken dargestellt sind.
Durch die Darstellung der Quantile können komplexere Verteilungen dargestellt werden. Die Verwendung des Mittelwertes und der Standardabweichung würden dagegen eine Normalverteilung voraussetzen. Zudem werden so maximale bzw. minimale Werte besser beachtet.

[1] Bei fünf Durchläufen entspricht jedes dieser Quantile genau einem Durchlauf.

| **Hyperparameter** | **Wert** |
|---|---|
| Aktivierung | ReLU ([Nair und Hinton, 2010]) |
| Batch-Größe | 200 |
| Optimierer | AdaDelta ([Zeiler, 2012]) |
| Max. Epochen | 5000 |
| Early-Stopping[4] | 2000 |

**Tabelle 5.1:** Tabelle der verwendeten Hyperparameter beim Training auf synthetischen Daten.
*Die hier dargestellte Tabelle beschreibt die verwendeten Hyperparameter beim Training aller Modelle, welche auf synthetischen Daten angewandt wurden.*

## 5.1 Synthetische Daten

Im Folgenden werden die Ergebnisse auf den in Abschnitt 4.1 (Seite 45) beschriebenen synthetischen Daten präsentiert. Hierzu gehört die Untersuchung einer passenden Modellarchitektur und die anschließenden Experimente zu den beiden Unsicherheitstypen (Modellunsicherheit und Datenunsicherheit). Zudem werden nach der Modellfindung einige Untersuchungen zur Verwendung von Dropout und der Größe des verwendeten Datensatzes aufgeführt.
Neben der Modellarchitektur müssen allerdings erst weitere Hyperparameter für das Training bestimmt werden. Diese Hyperparameter werden in Tabelle 5.1 beschrieben. Bei der Bestimmung dieser Hyperparameter gilt, dass diese entweder wenig Einfluss hatten (z.B. Batch-Größe[2]) oder absichtlich überschätzt (Max. Epochen, Early-Stopping) wurden[3]. Die verwendete Dropout-Rate ist nicht aufgeführt, da diese für die Untersuchungen variiert wird, um deren Auswirkungen auf die Verfahren zu testen. Dabei wurden die Dropout-Raten $p \in [0.05, 0.2, 0.5]$ verwendet, wobei $p = 0.5$ und $p = 0.2$ in der Praxis oft verwendete Standardwerte sind. Die Dropout-Rate von $p = 0.05$ wurde aus [Gal und Ghahramani, 2016b] übernommen. Dort wurde dieser Wert durch die Größe der verwendeten Modelle (zwei Hidden-Schichten mit jeweils 50 Neuronen) und

[2] Die Ergebnisse einer Versuchsreihe kann im Anhang A.1.5 eingesehen werden.
[3] Beispiele der Lernkurven zu den verwendeten Modellen können im Anhang A.1.1 eingesehen werden
[4] Falls sich nach 2000 Epochen der Fehler auf den Validierungsdaten nicht verbessert hat wird das Training beendet.

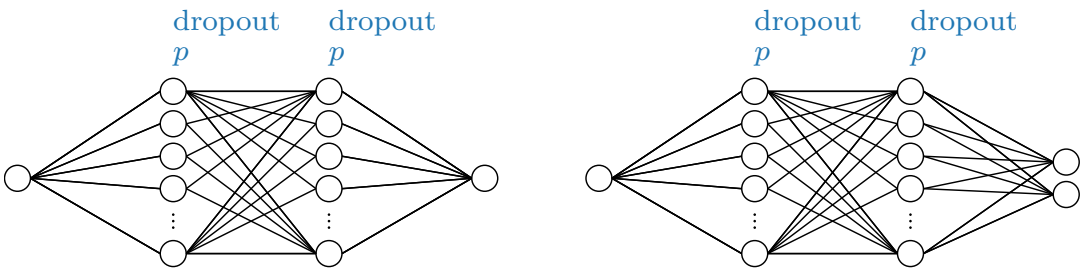
des Datensatzes begründet. Zudem stellt sie in dieser Arbeit einen Gegensatz zu den größeren Dropout-Raten dar und ermöglicht die Betrachtung der Auswirkungen kleiner Raten.
Für den Optimierungsalgorithmus wurde sich in dieser Arbeit für AdaDelta entschieden. Vorteil dieses Algorithmus ist, dass eine Parametrisierung der Lernrate erspart bleibt, da diese während des Trainings automatisch angepasst wird. Zudem sind die Optimierungsprobleme in dieser Arbeit einfach genug, sodass der verwendete Optimierungsalgorithmus keinen bedeutenden Einfluss auf das Ergebnis der Experimente haben sollte.

### 5.1.1 Modellarchitektur

Um die Verfahren zur Schätzung der Unsicherheiten anwenden zu können, wird zunächst eine geeignete Modellarchitektur benötigt. Für die Auswahl einer Modellarchitektur wurde sich auf Modelle mit zwei Hidden-Schichten beschränkt, welche die gleiche Anzahl an Neuronen haben. Für die Anzahl der Neuronen wurden die Werte $2, 4, 8, 16, 32, 64, 128, 256, 512$ und $1024$ getestet. Vorherige Experimente haben gezeigt, dass Modelle diese Größenordnung ausreichend sind, um das Problem der synthetischen Daten zu lösen[5]. Die Modelle mit sehr wenigen Neuronen werden verwendet, um den Verlauf der Unsicherheiten in Abhängigkeit der Modellgröße besser untersuchen zu können.
Für die Hidden-Schichten wurde eine ReLU-Aktivierung gewählt, da diese in heutigen Modellen häufig verwendet wird und die vorangegangen Untersuchungen ergaben, dass Modelle mit Sigmoid- oder TanH-Aktivierungen eine schlechtere Approximation der Daten liefern[6]. Für die Ausgabeschicht wurde, wie in der Regression üblich, eine lineare Aktivierungsfunktion gewählt. Für die Dropout-Rate wurde in diesem Experiment ein Wert von $p = 0.5$ gewählt, da dieser den größten Wert darstellt, welcher im Laufe dieser Arbeit untersucht wird. Die kleineren Dropout-Raten sollten daher kleinere Modelle erlauben, sodass in den folgenden Experimenten eine maximale Modellgröße ermittelt

[5] Auf eine Darstellung dieser Experimente wird für eine bessere Übersichtlichkeit verzichtet, zumal diese keine Schlussfolgerungen erlauben, welche nicht auch anhand der nachfolgenden Experimente gemacht werden können.

[6] Eine Diskussion der Auswirkungen der anderen Aktivierungsfunktionen ist im Anhang A.1.2 zu finden.




**(a)** N-Modell **(b)** K-Modell

**Abbildung 5.1:** Schema der Modellarchitektur für die synthetischen Daten.
*Zu sehen sind die Schemata für die beiden unterschiedlichen Modellarchitekturen für die synthetischen Daten. Links ist das varianzhomogene Modell und rechts das varianzheterogene Modell, welches eine explizite Schätzung der Datenunsicherheit vornimmt, abgebildet.*

werden kann.
Die Abbildung 5.1 zeigt die Schemata der resultierenden Modellarchitekturen. Links ist das varianzhomogene Modell abgebildet, welches von der Annahme eines von der Eingabe unabhängigen Rauschterms ausgeht. Das rechte Schema zeigt die Modellarchitektur nach [Kendall und Gal, 2017] zur Schätzung der Datenunsicherheit durch einen zusätzlichen Ausgang als varianzheterogenes Modell. Um eine Verwechslung der Datensätze und der Modelle zu vermeiden, werden im Folgenden die varianzhomogenen Modelle als N-Modelle (**N**ormale Modelle) und die varianzheterogenen Modelle als K-Modelle (**K**endall Modelle) bezeichnet.
Die Untersuchung der verschiedenen Modellarchitekturen wurde nur auf den varianzhomogenen Daten vorgenommen, da für beide Datensätze die generierende Funktion gleich ist. Ein komplexeres Modell hätte also für die varianzheterogenen Daten nur die Chance das Rauschen zu lernen und würde keine Verbesserung in der Abbildung der Kosinusfunktion darstellen.
Um die Ergebnisse besser bewerten zu können wurden zwei Baseline-Modelle definiert. Die Mittelwert-Baseline stellt eine minimale Qualität dar und ist definiert durch:

$$y_{Mittelwert}(x) = \mathbb{E}\Big[T\Big] \tag{5.1}$$

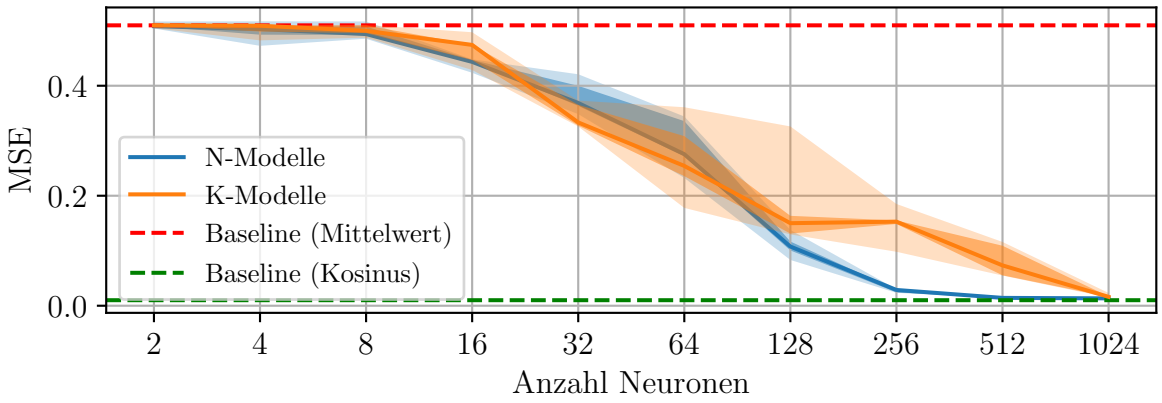



**Abbildung 5.2:** Verlauf des MSE in Abhängigkeit der Modellgröße.
*Zu sehen ist der Verlauf des erzielten MSE auf den Testdaten für die verschiedenen Modellgrößen. Die Dropout-Rate beträgt für jedes Modell $p = 0.5$. Abgetragen sind die Quantile über den fünf Durchläufen.*

wobei $T = \{t_1, t_2, \cdots, t_n\}$ die Menge der Zielwerte des Trainingsdatensatzes beschreibt. Dieses Modell entspricht einer Geraden auf Höhe des Mittelwertes dieser Zielwerte und jedes trainierte Modell sollte einen geringeren Fehler als dieses erzielen. Diese Baseline wird in den Abbildungen dieses Kapitels als rote gestrichelte Linie dargestellt.
Für eine obere Grenze der Modellgüte wird die Kosinus-Baseline definiert. Dieses Modell bildet die exakte Kosinusfunktion ab und ist daher definiert durch:

$$y_{Kosinus}(x) = \cos(x) \tag{5.2}$$

Ein trainiertes Modell kann nicht besser werden als die Kosinus-Baseline. Diese wird in folgenden Abbildungen durch eine grüne gestrichelte Linie repräsentiert.
In Abbildung 5.2 ist der Verlauf des Mean Squared Error (MSE) in Abhängigkeit der Modellgröße zu sehen. Zu erkennen ist, dass für eine Anzahl von bis zu 8 Neuronen alle Modelle nur wenig besser als die Mittelwert-Baseline sind. Ab einer Modellgröße von 256 Neuronen ändert sich die Qualität der N-Modelle nur noch sehr geringfügig für eine Verdoppelung der Neuronen. Auch sind die Ergebnisse für Modelle mit geringerem Fehler über die verschiedenen Durchläufe konsistent.
Für die K-Modelle ist scheinbar eine höhere Anzahl von Neuronen nötig, um das gleiche Problem zu lösen. Dies liegt daran, dass diese Modelle neben der Kosinusfunktion zusätzlich die Funktion der Datenunsicherheit abbilden müssen. Somit benötigen diese

mehr Parameter, um beide Funktionen lernen zu können. Auch ist ein großer Unterschied für Modelle mit 256 Neuronen zu erkennen. Was diesen starken Unterschied der Modelltypen verursacht ist allerdings nicht klar, besonders da die K-Modelle für eine Modellgröße von 32 Neuronen sogar einen kleineren MSE erreichen als N-Modelle. Zudem scheint der Einfluss der Initialisierung auf das Training bei den K-Modellen stärker als für die N-Modelle, da die Quantile dort einen größeren Abstand zueinander besitzen.
Für die weiteren Experimente wird auf Basis der hier erlangten Erkenntnisse festgehalten, dass eine Modellgröße von 256 Neuronen für die N-Modelle und 1024 Neuronen für die K-Modelle ausreichend ist. Für manche Untersuchungen werden dennoch Modelle mit 32 und 2048 Neuronen betrachtet, da diese Modelle darstellen, welche zu klein bzw. groß für die synthetischen Daten sind und das Verhalten der Unsicherheiten für diese Modelle von Interesse ist. Beispielsweise ist für die Datenunsicherheit zu prüfen, inwieweit die Qualität der Kalibrierung von der Modellkomplexität abhängt.

### No-U-Turn Sampler (NUTS)

Da für Modelle, deren Gewichte mittels des NUTS [Hoffman und Gelman, 2014] gesampelt wurden, keine Dropout-Schichten nötig sind und dieser Sampler auch bei einer Größe von 256 Neuronen pro Schicht eine lange Rechenzeit besitzt[7], werden für diesen kleinere Modelle verwendet. Als potentielle Modellarchitekturen wurden Modelle mit einer Hidden-Schicht und $5, 10, 15, 20, 25, 30, 60$ und 100 Neuronen betrachtet. Dabei wurde versucht die Modelle möglichst klein zu halten, um lange Laufzeiten zu vermeiden. Zudem hat sich schnell herausgestellt, dass für dieses Verfahren auch sehr kleine Modelle ausreichen sind, um das Problem der synthetischen Daten zu lösen.
Die Abbildung 5.3 zeigt den Verlauf des MSE in Abhängigkeit der Modellgröße. Deutlich erkennbar ist, dass ab einer Größe von 10 Neuronen der Fehler unverändert klein bleibt, während eine Größe von 5 Neuronen zu gering ist und einen deutlich größeren Fehler ergibt. Zudem sind die Ergebnisse über den Durchläufen weit konsistenter als für die N-Modelle oder die K-Modelle. Anzumerken ist, dass die durch den NUTS erreichten Fehler weit geringer sind als die der anderen Modelle.
Durch die Unterschiede in der Modellarchitektur und den dadurch resultierenden Ver-

[7] Ein Testdurchlauf mit zwei Schichten und 256 Neuronen hat eine Laufzeit von mehr als einem Monat.

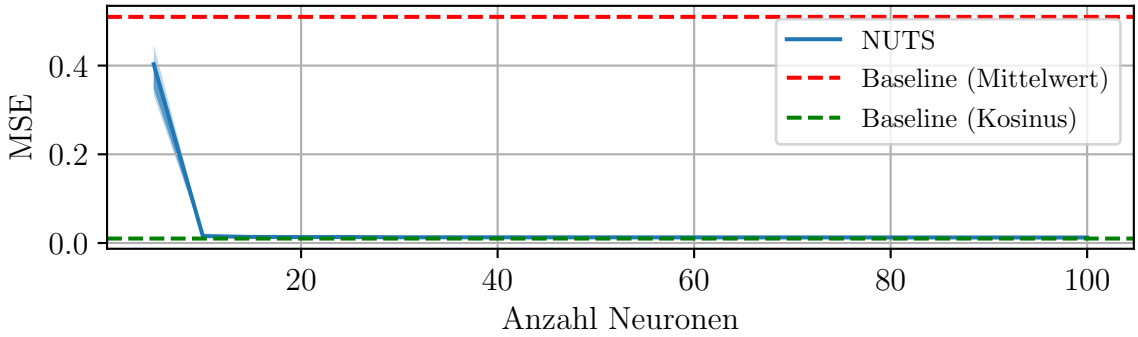



**Abbildung 5.3:** Verlauf des MSE in Abhängigkeit der Modellgröße für NUTS
*Zu sehen ist der Verlauf des MSE der Modelle, welche durch den NUTS gesampelt wurden, in Abhängigkeit der Anzahl an Neuronen in der Hidden-Schicht.*

teilungen über den Gewichten können die Unsicherheiten zwar nicht direkt mit den anderen Modellen verglichen werden, allerdings sollen diese auch nur eine Vergleichsbasis für Sachverhalte, wie das Verhalten bei unbekannten Daten, schaffen.

### Fazit

Als Fazit der Untersuchungen der Modellarchitektur kann festgehalten werden, dass K-Modelle eine höhere Komplexität als N-Modelle besitzen müssen, um eine vergleichbare Qualität der Approximation der Zielwerte zu erreichen. Auf Grund der Untersuchungen ergibt sich für die N-Modelle eine Anzahl von 256 Neuronen und für die K-Modelle von 1024 Neuronen als ausreichend.
Modelle, welche auf dem NUTS basieren, erreichen trotz ihrer geringeren Größe einen sehr geringen Fehler. Ein möglicher Faktor hierfür ist, dass diese Modelle kein Dropout verwenden und somit keine zusätzliche[8] Regularisierung verwendet wurde. Wichtig ist allerdings nur, dass die untersuchten Modellgrößen ausreichend sind, um diese in den folgenden Experimenten zu verwenden.

[8] Die Verwendung von Verteilungen und die Wahl der *A-priori-Verteilung* stellt bei bayesschen Verfahren eine Regularisierung dar.

### 5.1.2 Konvergenz der Modelle und deren Ausgaben

Nachdem nun eine geeignete Modellarchitektur für die synthetischen Daten gefunden wurde, können anhand dieser Modelle weitere Untersuchungen vorgenommen werden. Zum einen muss untersucht werden, wie viele Stichproben bei der Verwendung des MC-Dropout-Verfahrens nach [Gal und Ghahramani, 2016b] nötig sind, um eine ausreichend gute Schätzung des Mittelwertes und der Standardabweichung der Modellausgabe zu erhalten, da diese die eigentliche Schätzung des Zielwertes und deren Modellunsicherheit darstellen. Zum anderen kann untersucht werden, ob der generierte Datensatz groß genug ist, oder ob mehr Daten generiert werden müssen, damit der Fehler der Modelle nicht durch einen zu geringen Datensatz zustande kommt.

#### Schätzung der prädiktiven Verteilung

Für die Modellunsicherheiten, welche durch das MC-Dropout-Verfahren aus der Arbeit [Gal und Ghahramani, 2016b] berechnet werden, muss zunächst geklärt werden, wie viele Vorwärtsprozesse für deren Schätzung berechnet werden müssen. Dies ist wichtig, da sonst die geschätzten Momente der prädiktiven Verteilung keine guten Schätzungen ergeben. Für diese Untersuchung wurde das N-Modell mit 256 Neuronen verwendet, da sich diese Größe der Hidden-Schichten als ausreichend groß gezeigt hat und somit in der realen Anwendung verwendet werden würde. Bei dem verwendeten Daten handelt es sich um den varianzhomogenen Datensatz. Ziel dieser Untersuchung ist es eine gute Schätzung des Mittelwertes und der Standardabweichung der Modellausgabe zu erhalten. Das bedeutet, dass es ausreicht sich auf diese beiden Werte zu konzentrieren und zu vernachlässigen, inwieweit die generierten Stichproben die eigentliche Verteilung widerspiegeln.
Um die Konvergenz der Momente visuell darstellen zu können, wurde in Abbildung 5.4 jeweils die Summe der absoluten Änderungen der Werte über den gesamten Testdatensatz abgebildet. Das bedeutet, für den Fall der Konvergenz, zeigen die Kurven einen Wert von Null. Zu sehen sind dabei die Verläufe für den empirischen Mittelwert, der empirischen Standardabweichung und der Log-Likelihood basierend auf der Gauß-Verteilung. Letzteres kann zwar als Funktion der beiden Momente verstanden werden, dennoch wird dieses Maß aufgeführt, da die Log-Likelihood häufig zur Bewertung herangezogen wird.

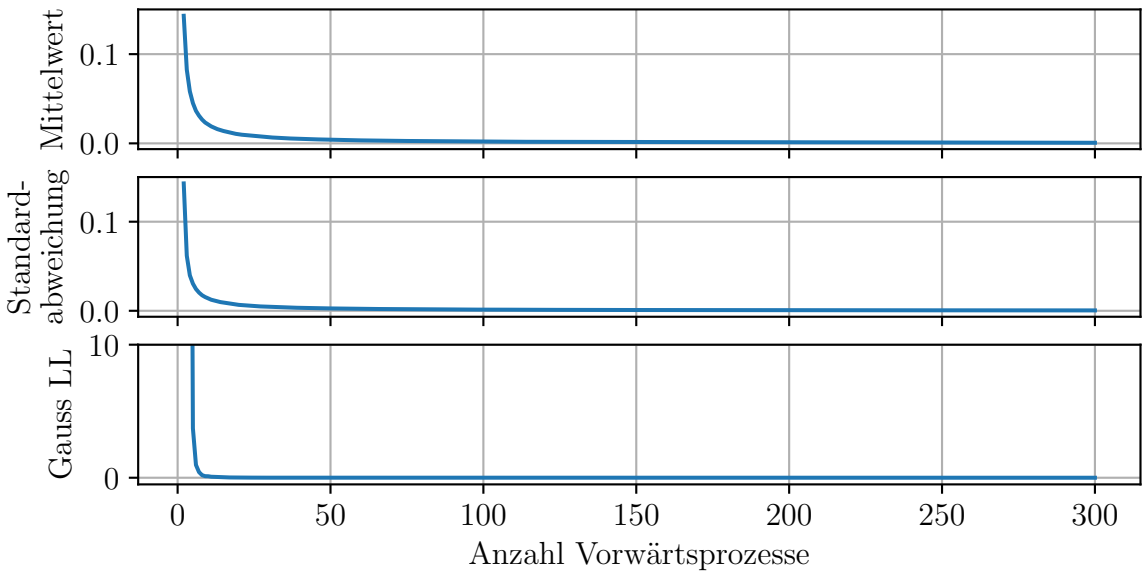



**Abbildung 5.4:** Verlauf der Änderung der Schätzungen des Mittelwertes, der Standardabweichung und der Gauß Log-Likelihood in Abhängigkeit der Anzahl an Vorwärtsprozessen

*Die Abbildung zeigt den Verlauf der summierten absoluten Änderung des Mittelwertes, der Standardabweichung und der Gauß Log-Likelihood in Abhängigkeit der Anzahl an Stichproben der Modellausgabe. Bei dem verwendeten Modell handelt es sich um das N-Modell mit* 256 *Neuronen und einer Dropout-Rate von* $p = 0.5$.

Wie in der Abbildung 5.4 zu sehen ist, wird die Änderung der Werte ab einer Stichprobenanzahl von 50 sehr gering. Dies bedeutet allerdings immer noch, dass für eine Berechnung der Ausgabe eines Modells für eine bestimmte Eingabe 50 Vorwärtsprozesse nötig sind. Dies bedeutet einen Faktor von 50 für den Rechenaufwand bei der Berechnung eines Modells. Da die hier verwendeten Modelle vergleichsweise klein sind, können diese Vorwärtsprozesse zwar parallel ausgewertet werden, allerdings für größere Modelle nicht mehr möglich. Für die Untersuchungen in [Gal und Ghahramani, 2016b] wurden 1000 Vorwärtsprozesse verwendet, allerdings haben die Autoren auch mit nur 10 Vorwärtsprozessen ausreichend gute Ergebnisse auf realen Daten erzielen können. Eine Anzahl von 100 Stichproben scheint, basierend auf diesen Tatsachen, für die Experimente in dieser Masterarbeit als angemessen, um sicherzustellen, dass eine ausreichend genaue Schätzung der Momente erreicht werden kann ohne unnötig viele Vorwärtsprozesse zu berechnen.

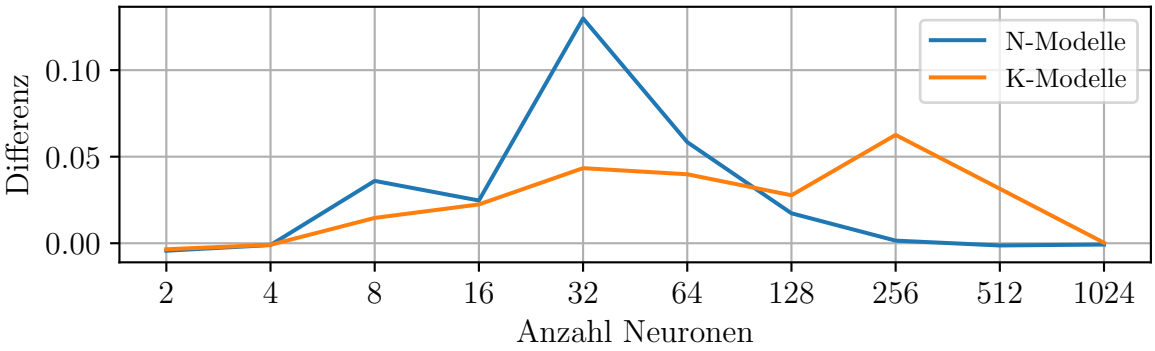



**Abbildung 5.5:** Unterschied der Fehler bei unterschiedlicher Schätzung des Mittelwertes

*Die Abbildung zeigt den Verlauf der Differenz der erreichten Fehlerwerte für beide Varianten der Mittelwertschätzung. Eine positive Differenz bedeutet dabei, dass die MC-Variante einen geringeren Fehler erreicht hat.*

### Vergleich der Ausgaben bei Verwendung von Dropout

Nachdem nun eine ausreichende Anzahl an Stichproben gefunden wurde, können nun weitere Untersuchungen zu der Verwendung von Dropout unternommen werden. Bei der Anwendung von Dropout werden, wie im Abschnitt 2.2 (Seite 7) beschrieben, normalerweise die Gewichte anhand der Dropout-Rate $p$ skaliert, sodass eine Schätzung des Mittelwerts der Ausgabe über alle Subnetze gebildet wird. Laut [Srivastava et al., 2014] gilt dies als praktikable Alternative zur Berechnung aller Subnetze welche ausreichend gute Ergebnisse liefert und einen deutlich geringeren Rechenaufwand besitzt. Durch die Verwendung des MC-Dropout-Verfahren [Gal und Ghahramani, 2016b] werden allerdings Stichproben der Subnetze benötigt. Das bedeutet, dass eine Schätzung des Mittelwertes der Modellausgabe durch diese Stichproben möglich ist, ohne zusätzlichen Rechenaufwand zu verursachen[9]. Deshalb ist es von Interesse zu untersuchen, welche Auswirkungen die beiden Varianten auf die Evaluation der Modelle haben.

Hierfür wurde der MSE für beide Arten der Schätzung des Mittelwertes und unterschiedliche Modellgrößen untersucht. Abbildung 5.5 zeigt einen Vergleich der beiden Methoden anhand des erreichten MSE. Abgetragen ist der Verlauf der Differenz zwischen den

[9] Die Berechnung des Mittelwertes über den Ausgaben kann hierbei vernachlässigt werden

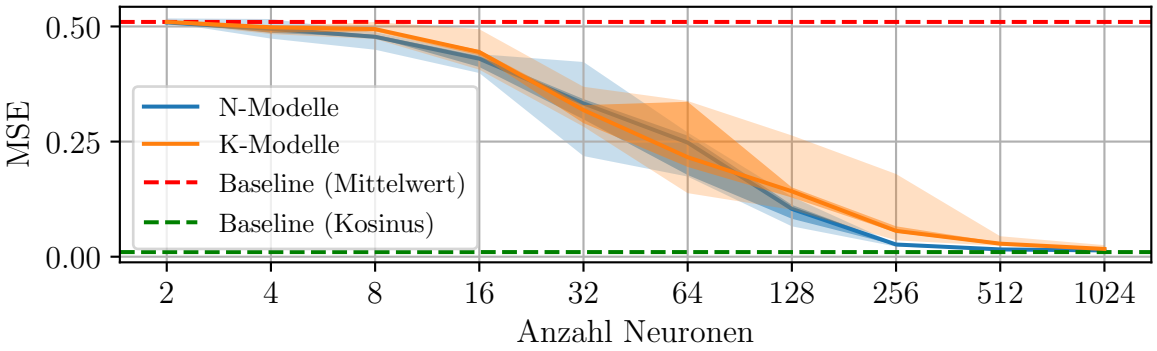



**Abbildung 5.6:** Verlauf des MSE in Abhängigkeit der Modellgröße für die Mittelwertberechnung anhand der MC-Samples
*Zu sehen ist der Verlauf des MSE bei Schätzung des Mittelwertes anhand der MC-Samples für beide Modelltypen und verschiedene Modellgrößen.*

beiden erreichten Fehlerwerten für den jeweils besten Durchlauf[10]. Ein positiver Wert der Differenz bedeutet dabei, dass die Schätzung über die Skalierung der Gewichte einen größeren Fehler ergeben hat. Grundsätzlich ist zu erkennen, dass der MSE besonders für die mittleren Modellgrößen für die Mittelwertbildung durch die Stichproben kleiner ist als für die Skalierung. Auch wird die Differenz kaum negativ, was bedeutet, dass die Schätzung durch Skalierung der Gewichte nie deutlich besser ist. Diese Ergebnisse decken sich mit der Arbeit [Srivastava et al., 2014] in der die Skalierung der Gewichte nur als praktikable Vereinfachung zur Berechnung aller Subnetze eingeführt wurde. Dabei werden zwar schlechtere Ergebnisse erzielt, allerdings werden diese für die praktische Anwendung als ausreichend beschrieben.
Zusätzlich zeigt sich ein großer Unterschied für K-Modelle bei einer Anzahl von 256 Neuronen. Für diese Netzwerkgröße zeigte sich zuvor in Abbildung 5.2 (Seite 63) ein starker Unterschied der Fehler zwischen N-Modellen und K-Modellen. Abbildung 5.6 zeigt nun den Verlauf des Fehlers der beiden Modelltypen unter Verwendung des Mittelwertes über den Stichproben. Für diese Art der Mittelwertbildung gleichen sich die beiden Kurven stark und der zuvor beobachtete Unterschied der Modelle mit 256 Neuronen ist nicht mehr zu erkennen. Dies ändert allerdings nichts an der Tatsache, dass K-Modelle komplexer sein müssen als N-Modelle.
Für alle folgenden Experimente wird anhand der hier gemachten Beobachtungen nur

[10] Gemessen am MSE durch Skalierung der Gewichte.

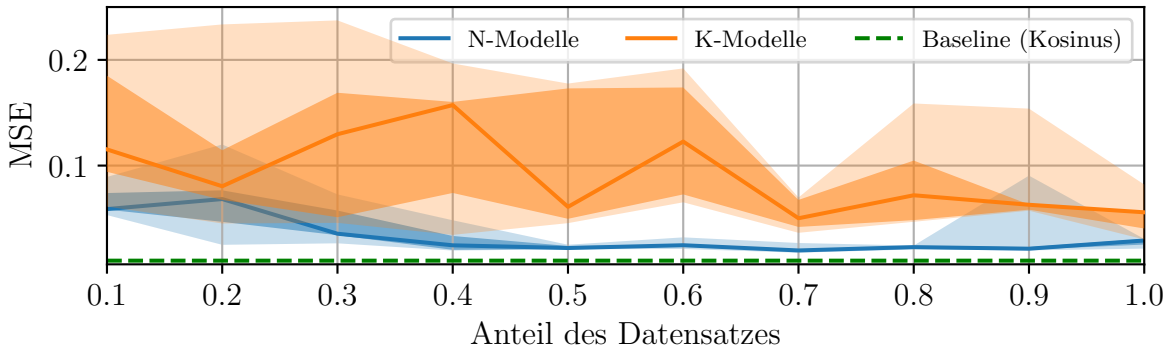



**Abbildung 5.7:** Verlauf des Fehlers in Abhängigkeit der Datengröße
*Abgebildet ist der Verlauf des MSE in Abhängigkeit der Datengröße. Verwendet wurde hierfür der varianzhomogene Datensatz und eine Dropout-Rate von* $p = 0.5$.

noch der MSE auf Basis der Mittelwertschätzung durch der Stichproben betrachtet.

### Datensatzgröße

Da für den synthetischen Datensatz neue Daten ohne großen Aufwand generiert werden können, lohnt es sich zu untersuchen, ob der bisher beschriebene Datensatz ausreichend groß ist. Für diese Untersuchungen wurde der Verlauf des Fehlers in Abhängigkeit der Datensatzgröße betrachtet. Das bedeutet, dass jeweils nur ein Teil des gesamten Trainingsdatensatzes verwendet wurde, während der Validierungs- bzw. der Testdatensatz identisch geblieben sind. Für jeden der fünf durchgeführten Durchläufe unterscheidet sich damit auch der Trainingsdatensatz, da dieser jeweils zufällig zusammengestellt wurde. Dieses Vorgehen könnte allerdings zu einer erhöhten Varianz über den Durchläufen führen, da die unterschiedlichen Trainingsdatensätze unterschiedliche Informationen enthalten können. Für den Fall, dass eine ausreichende Anzahl an Daten vorliegt, sollte der Fehler dennoch konvergieren, sodass eine Ermittlung einer Mindestgröße des Trainingssets möglich ist.
In Abbildung 5.7 ist nun der Verlauf des MSE in Abhängigkeit der Datensatzgröße zu sehen. Für die Hidden-Schichten wurden, nach den Ergebnissen aus dem Abschnitt 5.1.1 (Seite 61), jeweils 256 Neuronen verwendet, da sich diese Zahl als ausreichend für die N-Modelle herausgestellt hat. Für einen besseren Vergleich wurden auch für die K-Modelle 256 Neuronen verwendet, da die Anzahl der benötigten Trainingsdaten meist abhängig von der Modellgröße ist. Für die Dropout-Rate wurde der Wert $p = 0.5$

gewählt, da dieser den größten Wert darstellt, der in dieser Arbeit verwendet wurde. Somit formuliert dieser das schwierigste Optimierungsproblem und kleinere Dropout-Raten würden erwartungsgemäß weniger Daten benötigen[11].
Zu erkennen ist, dass der MSE für die N-Modelle mit zunehmender Datengröße abnimmt und sich ab einer Datengröße von 50% nur noch geringfügig ändert. Das bedeutet, dass für diese Modelle die Hälfte der Trainingdaten ausreichen würde. Auch ist zu erkennen, dass der MSE für die kleinsten Datengrößen stärker schwankt. Dies zeigt, dass in den zufällig gezogenen Teildatensätzen unterschiedliche Informationen stecken. Erst ab 50% der Daten sind alle nötigen Informationen im Datensatz vorhanden, damit das Modell die Kosinusfunktion ausreichend abbilden kann.
Für die K-Modelle unterscheidet sich der Verlauf stark von dem der N-Modelle. Hier fällt der Fehler erst für 70% der Daten merklich, wobei auch die Varianz über den Durchläufen geringer wird. Dies lässt sich vermutlich damit erklären, dass die K-Modelle zusätzlich die Funktion des Rauschterms aus den Daten ableiten müssen und somit mehr Daten benötigen, um diese zusammen mit der Kosinusfunktion abbilden zu können. Auch müssen den K-Modellen möglicherweise mehr Daten zur Verfügung gestellt werden, damit diese ihre Fehler nicht durch Datenunsicherheit erklären, sondern versuchen eine bessere Schätzung der eigentlichen Funktion zu erreichen.
Zudem ist sehr deutlich erkennbar, dass der MSE für die K-Modelle stärker schwankt als für die N-Modelle. Dies resultiert vermutlich aus der Tatsache, dass die K-Modelle die Möglichkeit haben ihre Fehler durch Datenunsicherheit zu erklären. Für den Fall, dass nicht genug Daten zur Verfügung stehen, um die eigentliche Funktion und die Funktion des Rauschterms zu lernen, kann es nun vorkommen, dass das Modell in manchen Bereichen eine sehr einfache Funktion lernt und die resultierenden Fehler durch die Datenunsicherheit erklärt. Da sich die Trainingsdaten für jeden Durchlauf unterscheiden tritt dieser Fall möglicherweise unterschiedlich häufig auf, sodass für manche Durchläufe ausreichend Daten in allen Bereichen der Eingabe vorhanden sind.
Eine Verbesserung des Fehlers bei einem noch größeren Datensatz ($> 100\%$) ist nicht zu erwarten. Eine solche Verbesserung des MSE nachdem dieser, wie in der Abbildung ersichtlich, konvergiert ist, würde bedeuten, dass bei einem noch größeren Datensatz neue informative Beispiele hinzukamen. Bei dem hier verwendeten Datensatz kann allerdings aufgrund der geringen Komplexität des Problems, angenommen werden, dass

[11] Im Anhang A.1.6 sind die Ergebnisse für kleinere Dropout-Raten einsehbar.

der gewählte Trainingsbereich ausreichend abgetastet wurde.
Für eine Bewertung der Datensatzgröße ist anzumerken, dass der verwendete Rauschterm und die Likelihood zur Berechnung des Fehlers beim Training der K-Modelle auf der gleichen Verteilung (Normalverteilung) basieren. Da dies möglicherweise einen Idealfall für die Optimierung darstellt, können Ergebnisse auf realen Datensätzen möglicherweise anders aussehen. Dennoch kann für die synthetischen Daten festgehalten werden, dass für das Training der K-Modelle mehr Daten nötig zu sein scheinen als für die N-Modelle. Zudem scheint der generierte Datensatz groß genug, da für beide Modelltypen der Fehler bei 100% des Datensatzes konvergiert ist.

**Fazit**

In diesem Abschnitt wurde gezeigt, dass eine Auswertung von Modellen anhand der Stichproben des Verfahrens nach [Gal und Ghahramani, 2016b] vorteilhaft ist. Zudem haben sich 100 Vorwärtsprozesse als ausreichend für die Schätzung der Momente der prädiktiven Verteilung herausgestellt.
Die Untersuchungen der Datensatzgröße haben ergeben, dass die Anzahl an generierten Datenpunkten ausreichend ist. Zudem benötigt das Training von K-Modellen mehr Daten als das der N-Modelle mit gleicher Größe.

### 5.1.3 Modellunsicherheit

Nachdem eine Basis für die Untersuchungen der verschiedenen Unsicherheiten geschaffen wurde, wird nun die Modellunsicherheit, welche anhand des MC-Dropout-Verfahrens nach [Gal und Ghahramani, 2016b] berechnet wird, untersucht. Dabei werden die Ergebnisse mit denen des NUTS [Hoffman und Gelman, 2014] verglichen, um herauszufinden, ob sich die Modellunsicherheiten ähnlich verhalten. Ein expliziter Vergleich der Werte für die Unsicherheiten ist dabei nicht möglich, da die unterschiedlichen Modellarchitekturen eine unterschiedliche Verteilung über den Gewichten und damit eine unterschiedliche Modellunsicherheit ergeben.
Die folgenden Experimente können dabei in zwei Teile unterteilt werden. Zuerst wird die Auswirkung von Modellgröße und Dropout-Rate auf die Schätzung der Modellunsicherheit untersucht. Diese Untersuchungen sind von Bedeutung, da Modellarchitektur und Dropout-Rate in der Praxis nicht auf Basis der Modellunsicherheit bestimmt

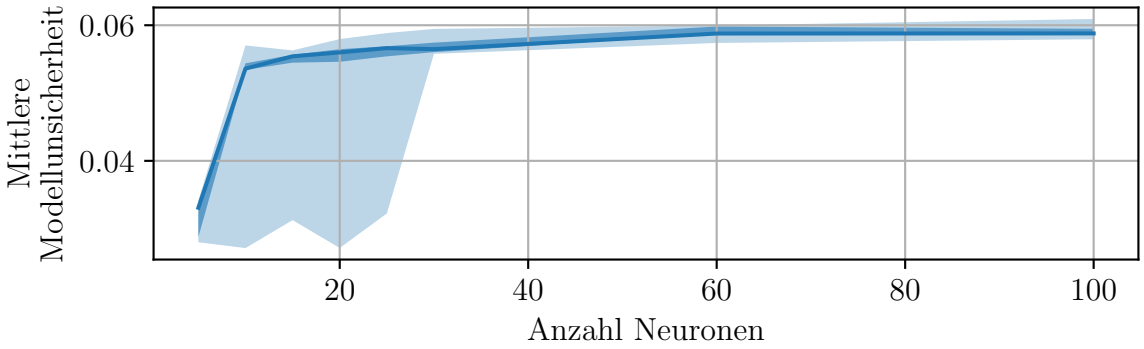
werden. Daher muss dem Anwender bewusst sein, wie sich seine Wahl auf die geschätzte Modellunsicherheit auswirkt. Für diese Untersuchungen wird jeweils die mittlere Modellunsicherheit eines Modells über dem gesamten Testdatensatz betrachtet. Das bedeutet, dass die unbekannten Bereiche hier zuerst vernachlässigt und nur Daten aus dem Trainingsbereich betrachtet werden.
Anschließend werden die Modellunsicherheiten im Anwendungsfall der Detektion von unbekannten Daten getestet. Diese Untersuchungen bilden eine Grundlage für die späteren Experimente auf dem Datensatz der SES, welche im Abschnitt 5.2.3 (Seite 107) beschrieben werden. Zudem stellt die Detektion von unbekannten Daten einen der Hauptanwendungsfälle für die Verwendung von Modellunsicherheit dar. Für diese Untersuchungen werden die mittleren Modellunsicherheiten über den einzelnen Eingabebereichen, welche im Abschnitt 4.1 (Seite 45) beschrieben wurden, betrachtet. Für die Untersuchung der Kalibrierung ist die Modellunsicherheit, im Gegensatz zur Datenunsicherheit, nicht geeignet[12].

### Auswirkungen der Modellgröße

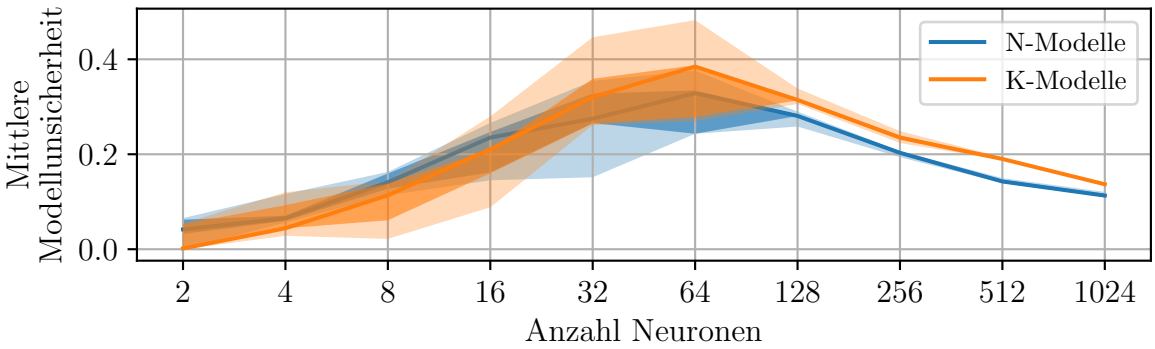
Für die Untersuchung der Auswirkungen der Modellgröße auf die geschätzten Modellunsicherheiten wurden die im Abschnitt 5.1.1 (Seite 61) untersuchten Modelle betrachtet. Dabei wurde sich auf die mittlere Modellunsicherheit konzentriert, welche den Mittelwert der Modellunsicherheiten über dem gesamten Testdatensatz beschreibt.
Um einen Vergleichswert für die resultierenden Modellunsicherheiten zu erhalten, ist in Abbildung 5.8a der Verlauf der Modellunsicherheiten für den NUTS bei unterschiedlicher Modellgröße zu sehen. Erkennbar ist, dass die Modellunsicherheit für eine Anzahl von 5 Neuronen deutlich kleiner ist als für größere Modelle. Ab 10 Neuronen stabilisiert sich die Modellunsicherheit für die restlichen Modelle. Für Modelle mit bis zu 30 Neuronen gab es allerdings jeweils einen Durchlauf, bei dem die Modellunsicherheit ähnlich klein geschätzt wurde, wie bei Modellen mit 5 Neuronen. Dabei ist anzumerken, dass dieser Unterschied in den Durchläufen anhand des MSE nicht erkennbar war.
Die Abbildung 5.8b zeigt nun den Verlauf der mittleren Modellunsicherheit, welche durch das MC-Dropout-Verfahren geschätzt wurde, als Funktion der Modellgröße. Auffällig ist der unterschiedliche Kurvenverlauf zur Schätzung durch den NUTS. Der

[12] Eine detailliertere Erläuterung dazu ist im Anhang A.1.3 zu finden.

Verlauf der Modellunsicherheit zeigt hier einen Wendepunkt bei einer Modellgröße von 64 Neuronen. Ab dieser Modellgröße nimmt die Schätzung der Modellunsicherheit mit der Anzahl der Neuronen ab. Allerdings könnte dieser Wendepunkt, bedingt durch die Wahl der untersuchten Modellgrößen, in Abbildung 5.8a versteckt sein. Die Tatsache, dass für beide Kurven ersichtlich ist, dass kleine Modelle eine gewisse Varianz über den Durchläufen besitzen lässt vermuten, dass sich dieser Wendepunkt für die NUTS-Modelle bei einer Anzahl von 60 Neuronen befindet. Zudem scheint die Modellunsicherheit für Modelle mit mehr als 60 Neuronen leicht abzufallen.
Für die N-Modelle ist der Kurvenverlauf identisch zu den K-Modellen. Allein die




**(a)** Verlauf für NUTS.




**(b)** Verlauf für N-Modelle und K-Modelle.

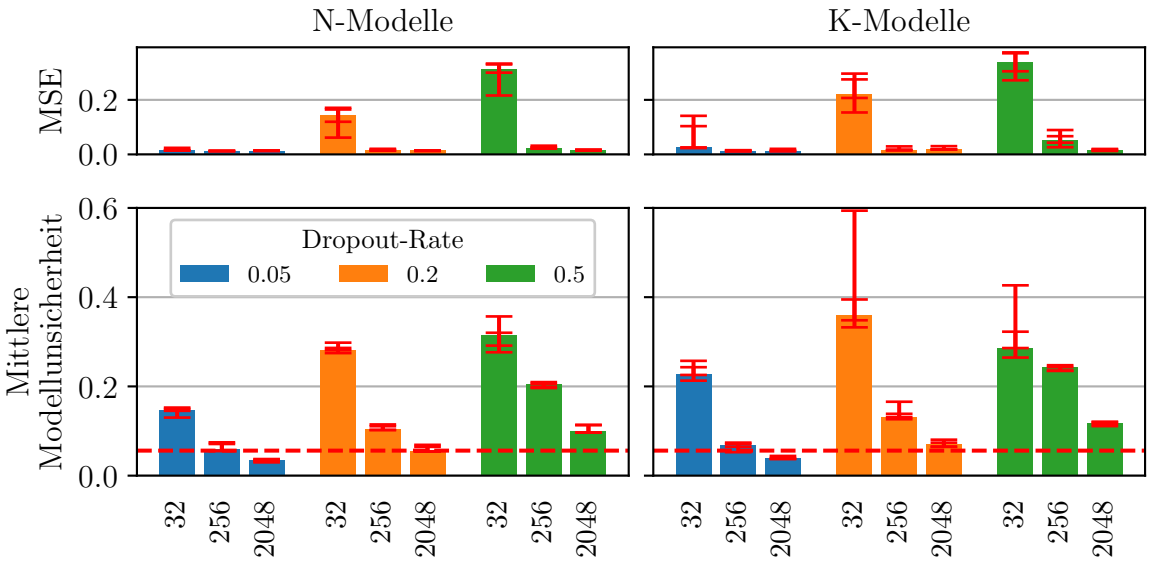
**Abbildung 5.8:** Verlauf der mittleren Modellunsicherheit in Abhängigkeit der Modellgröße

*Zu sehen ist jeweils der Verlauf der mittleren Modellunsicherheit für NUTS-Modelle, N-Modelle und K-Modelle als Funktion der Modellgröße.*

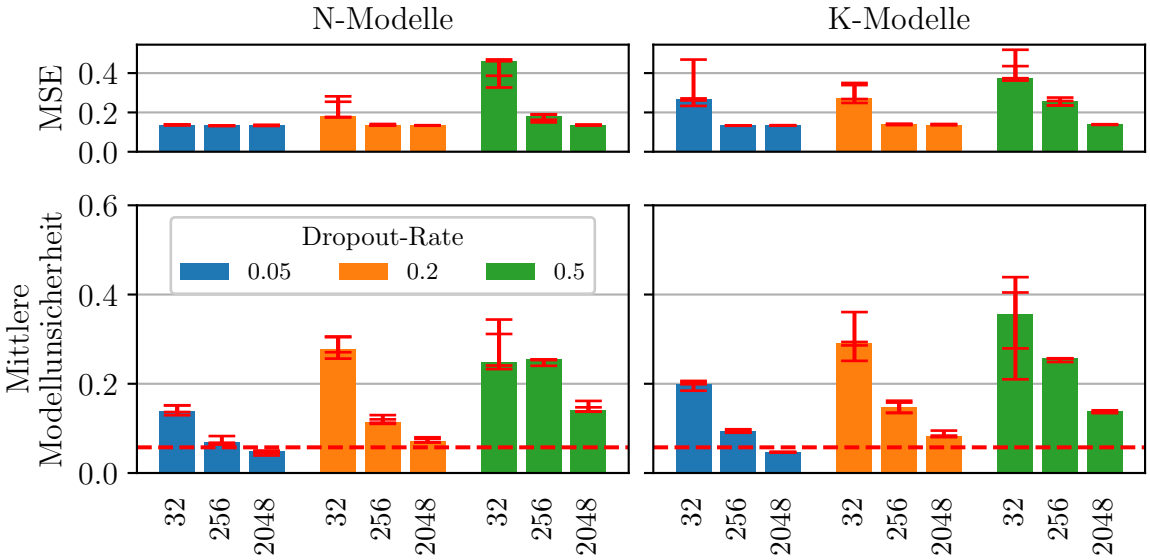
Schätzungen der Modellunsicherheit bei einer größeren Anzahl an Neuronen ist für K-Modelle etwas größer als die der N-Modelle. Dies lässt sich vermutlich wieder durch die Tatsache erklären, dass K-Modelle zusätzlich eine Funktion für die Datenunsicherheit lernen müssen. Dadurch könnte eine erhöhte Unsicherheit in der Parameterschätzung entstehen, da sich die verfügbaren Parameter auf beide Funktionen aufteilen müssen. Den N-Modellen hingegen stehen alle Parameter für die Abbildung der Kosinusfunktion zur Verfügung.

### Auswirkungen der Dropout-Rate

Neben der Modellgröße ist die Auswirkung der Dropout-Rate auf die Modellunsicherheit von Interesse. Dass eine Abhängigkeit der Modellunsicherheiten von der Dropout-Rate herrscht wurde bereits in [Osband et al., 2018] erläutert. Wie sich diese Abhängigkeit genau auswirkt soll in diesem Abschnitt untersucht werden. Da die Auswirkung der Modellgröße bereits betrachtet wurde, werden in diesem Abschnitt nur Modelle mit $32, 256$ und 2048 Neuronen betrachtet. Allerdings werden dafür zusätzlich die Ergebnisse auf den varianzheterogenen Daten ausgewertet.
Die Abbildung 5.9 zeigt nun den Verlauf der mittleren Modellunsicherheit auf den Testdaten in Abhängigkeit der Dropout-Rate. Wie erwartet ist generell eine steigende Modellunsicherheit mit zunehmender Dropout-Rate zu erkennen. Ausnahme bildet allerdings der Verlauf mancher Modelle mit 32 Neuronen. Für die N-Modelle auf varianzheterogenen Daten und für die K-Modelle auf varianzhomogenen Daten ist die Modellunsicherheit bei dieser Modellgröße bei einer Dropout-Rate von $p = 0.5$ geringer als für $p = 0.2$.
Dies könnte ein Resultat der geringen Anzahl an Versuchswiederholungen sein, da sich die abgebildeten Quantile bei einer größeren Anzahl möglicherweise noch verändern könnten. Allerdings kann das Abfallen der Modellunsicherheit auch durch die geringe Modellkomplexität der Modelle mit 32 Neuronen begründet werden. Die geringe Modellgröße und die hohe Dropout-Rate könnten dazu führen, dass eine stark vereinfachte Funktion gelernt wird, wobei die Unsicherheit der dafür benötigten Parameter geringer ist, da weniger Freiheitsgrade zur Verfügung stehen, um Funktionen ähnlicher Qualität zu erzeugen. Somit erzeugen alle betrachteten Subnetze des Modells sehr ähnliche und sehr einfache Funktionen.



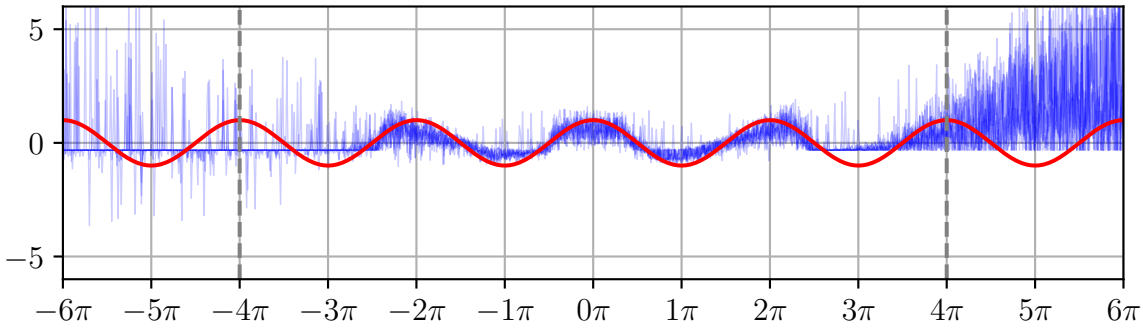

**(a)** Verlauf auf varianzhomogenen Daten.



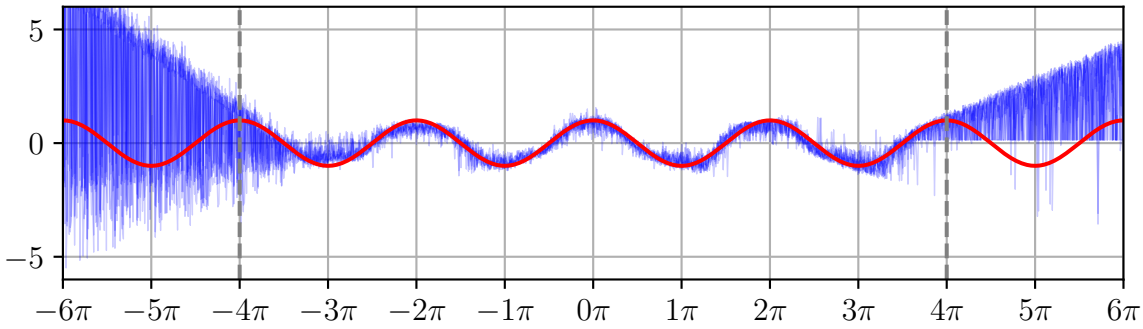

**(b)** Verlauf auf varianzheterogenen Daten.

**Abbildung 5.9:** Mittleren Modellunsicherheit in Abhängigkeit der Dropout-Rate
*Zu sehen sind die mittleren Modellunsicherheit in Abhängigkeit der Dropout-Rate für verschiedene Datensätze (oben: varianzhomogen, unten: varianzheterogen), Modelltypen (links: N-Modelle, rechts: K-Modelle) und -größen. Die roten gestrichelten Linien beschreiben die mittleren Modellunsicherheiten der NUTS-Modelle. Zusätzlich ist für alle N-Modelle bzw. K-Modelle der MSE angegeben.*

Um diese These zu überprüfen sind in Abbildung 5.10 jeweils 20 Vorwärtsprozesse für Modelle mit 32 Neuronen und einer Dropout-Rate von $p = 0.5$ bzw. $p = 0.2$ abgebildet. Anhand dieser Abbildung ist zu erkennen, dass das Modell mit $p = 0.5$ besonders im negativen Bereich der Eingabe eine horizontale Gerade darstellt. Der Unterschied zum positiven Eingabebereich ist vermutlich auf die unterschiedliche Datendichte zurückzuführen, da im negativen Bereich weniger Trainingsdaten vorhanden sind als im positivem Bereich. Diese Beobachtung belegt die oben getroffene These, dass die Modelle mit 32 Neuronen bei einer hohen Dropout-Rate einfache Funktionen mit wenig Modellunsicherheit lernen.




**(a)** Modell mit 32 Neuronen.




**(b)** Modell mit 256 Neuronen.

**Abbildung 5.10:** Gelernte Funktionen der Modelle

*Abgebildet sind jeweils 20 Vorwärtsprozesse der beiden N-Modelle mit 32 bzw. 256 Neuronen auf varianzheterogenen Daten. Die einzelnen Vorwärtsprozesse sind dabei übereinandergelegt. Die rote Kurve zeigt die zu lernende Kosinusfunktion.*

**Erkennbarkeit von unbekannten Daten**

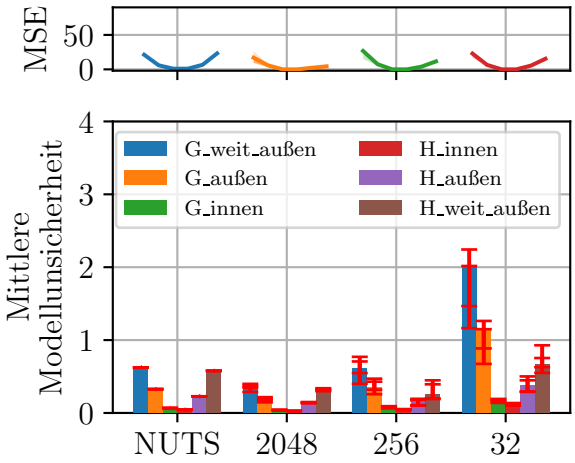
Eine der Hauptanwendungen der Modellunsicherheit ist die Erkennung von unbekannten Daten. Dadurch kann zum Beispiel erkannt werden, falls sich neue Datenpunkte stark vom Trainingsdatensatz unterscheiden, was zum Beispiel zur Folge hätte, dass das Modell in diesem Falle extrapoliert. Deshalb wird im Folgenden untersucht, inwieweit die geschätzten Modellunsicherheiten der verschiedenen Methoden für diese Aufgabenstellung geeignet sind. Dabei werden die Unsicherheiten anhand der in Abschnitt 4.1 (Seite 45) definierten Bereiche untersucht und verglichen. Ziel dabei ist es, die Daten in den unbekannten Bereichen anhand der Modellunsicherheit zu erkennen und von den bekannten Daten zu unterscheiden. Die folgenden Untersuchungen dienen zudem als Basis für die Experimente auf den Daten der SES, welche im Abschnitt 5.2 (Seite 91) präsentiert werden.
Da sich aus dieser Aufgabenstellung ein Klassifikationsproblem ergibt, können Qualitätsmaße aus der Klassifikation zur Bewertung verwendet werden. Speziell werden in dieser Arbeit die ROC-Kurven genutzt, um zu entscheiden, welche Verfahren am besten dafür geeignet sind unbekannte Daten anhand der Modellunsicherheit zu erkennen.
In den Abbildungen 5.11 und 5.12 ist eine Übersicht der geschätzten Modellunsicherheiten auf den unterschiedlichen Bereichen für N-Modelle und K-Modelle abgebildet. Entlang der x-Achse sind, neben dem NUTS-Modell die Modellgrößen abgetragen. Für die Ergebnisse des NUTS wurde repräsentativ das Modell mit 30 Neuronen verwendet, da kaum ein Unterschied im MSE und der Modellunsicherheit zu größeren Modellen bestand. Die Balken beschreiben die mittlere Modellunsicherheit auf den jeweiligen Bereichen und die darüber liegenden Kurven beschreiben den dazugehörigen MSE. Für jede Zeile in den Abbildungen wurde jeweils eine unterschiedliche Dropout-Rate verwendet. In der linken Spalte wurde der varianzhomogene Datensatz und in der rechten Spalte der varianzheterogene Datensatz verwendet. In jeder Zeile wurden die Ergebnisse des selben NUTS-Modells wiederholt, da bei diesen kein Dropout verwendet wird und die Dropout-Rate damit keinen Einfluss auf diese Modelle hat. Das bedeutet es sind insgesamt zwei verschiedene NUTS-Modelle abgebildet, für jeden Datentyp eines.

**N-Modelle** Für die Modellunsicherheiten ist zu erkennen, dass diese generell mit steigender Dropout-Rate für alle Bereiche und Datensätze ansteigen. Allerdings gibt es auffällige Ausnahmen für die Unsicherheiten auf unbekannten Daten. Die Modellunsicherheiten auf den varianzheterogenen Daten und der Modelle mit einer Anzahl von 32 Neuronen pro Schicht in Abbildung 5.11d und 5.11f verhalten sich genau entgegen dieser Beobachtung. Für eine Dropout-Rate von $p = 0.2$ sind die Modellunsicherheiten auf den beiden unbekannten Bereichen G_außen und G_weit_außen größer als bei einer Dropout-Rate von $p = 0.5$. Diese Beobachtung ist analog zu den Ergebnissen der Untersuchung der Auswirkungen der Dropout-Rate. Der Grund hierfür ist daher wieder, dass kleine Modelle bei hoher Dropout-Rate eine einfache Funkion lernen (z.B. eine Gerade auf Höhe des Mittelwertes) während die Unsicherheit in den Parametern für diese Funktionen sehr gering ist. Anhand des Verlaufs des MSE in der Abbildung 5.11f ist zusätzlich erkennbar, dass die Modelle eine Funktion nahe der Geraden um den Mittelwert gelernt haben, da diese einen vergleichsweise geringen Fehler auf den unbekannten Daten erzielen. Dies gilt allerdings nur, da die verwendete Kosinusfunktion periodisch ist und Modelle mit einer konstanten Funktion auch in unbekannten Bereichen nahe der Zielwerte liegen[13].

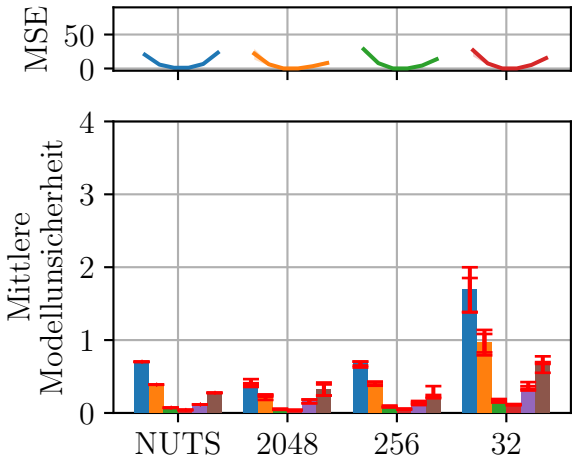
Die Annahme, dass die Modellunsicherheit auf den Bereichen G_weit_außen und G_außen größer sein sollte als die Unsicherheiten auf H_außen und H_weit_außen, da diese an die Trainingsbereichen mit geringerer Datendichte angrenzen, kann anhand dieser Ergebnisse nicht bestätigt werden. Für Modelle mit 2048 und 32 Neuronen lassen sich Beispiele finden, bei denen der umgekehrte Fall zutrifft. Da dies allerdings erst bei einer Dropout-Rate von 0.2 auftritt, kann dies ein Indiz dafür sein, dass dieser Wert für das hier untersuchte Problem ungeeignet ist. Folglich wäre die richtige Wahl der Dropout-Rate von großer Bedeutung für eine gute Schätzung der Modellunsicherheit.

Die Modellunsicherheit der Modelle mit 256 Neuronen zeigt grundsätzlich das erwartete Verhalten. Sie steigt mit der Dropout-Rate an und die Dichte der Trainingsdaten scheint einen Einfluss auf die Unsicherheiten in den angrenzenden unbekannten Bereichen zu haben. Da diese Modellgröße als ausreichend für die N-Modelle gilt, kann dies bedeuten, dass für Modelle, welche so groß wie nötig aber so klein wie möglich sind, das MC-Dropout Modellunsicherheiten liefert, welche ein erwartetes Verhalten haben.

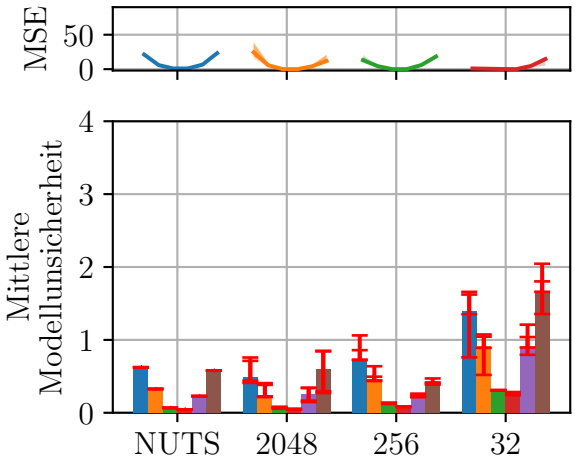
[13] Für eine Darstellung der gelernten Funktion für Modelle mit 32 Neuronen siehe Abbildung 5.10 (Seite 77).



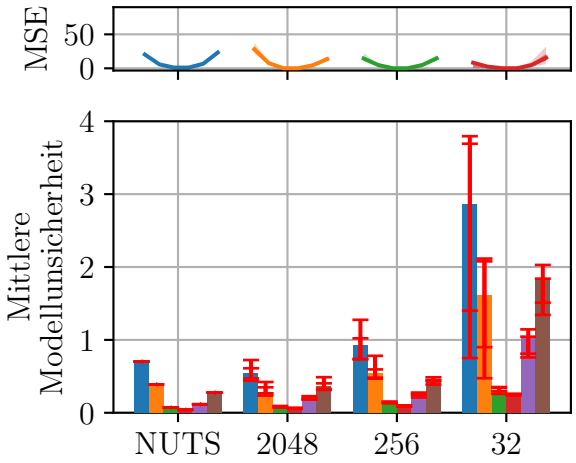

**(a)** Dropout-Wkt. $p = 0.05$, varianzhomogen



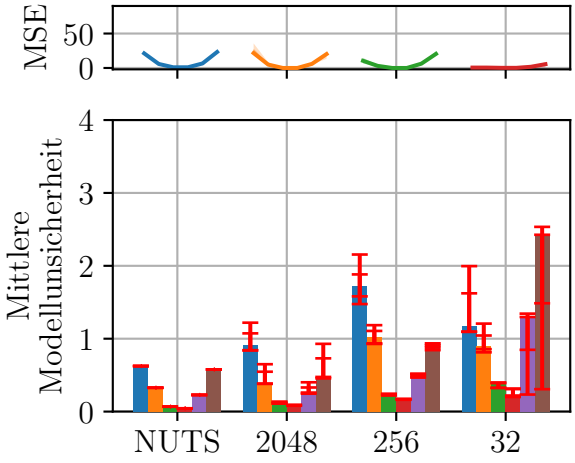

**(b)** Dropout-Wkt. $p = 0.05$, varianzheterogen



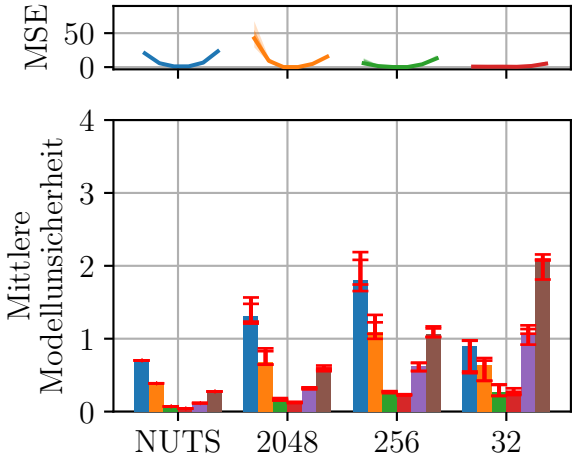

**(c)** Dropout-Wkt. $p = 0.2$, varianzhomogen




**(d)** Dropout-Wkt. $p = 0.2$, varianzheterogen

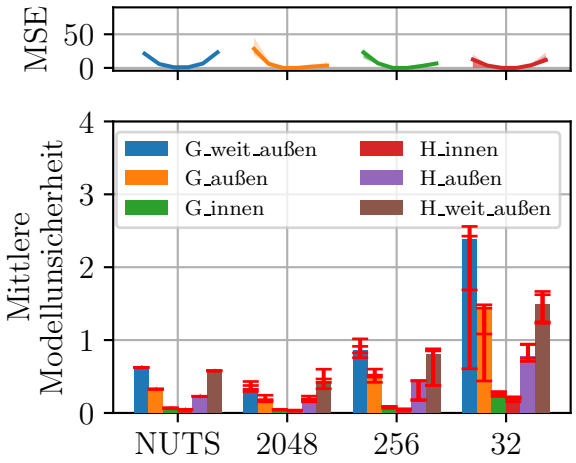



**(e)** Dropout-Wkt. $p = 0.5$, varianzhomogen

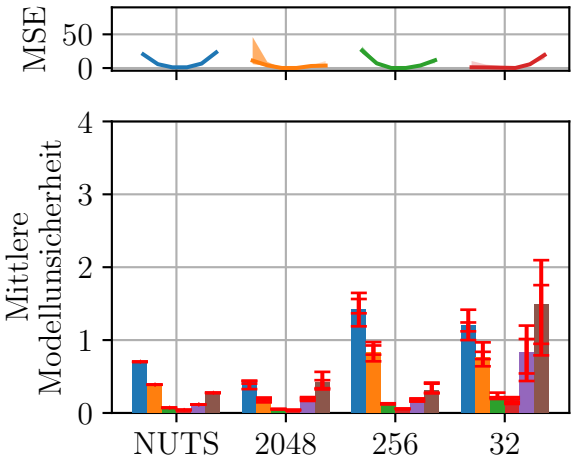



**(f)** Dropout-Wkt. $p = 0.5$, varianzheterogen

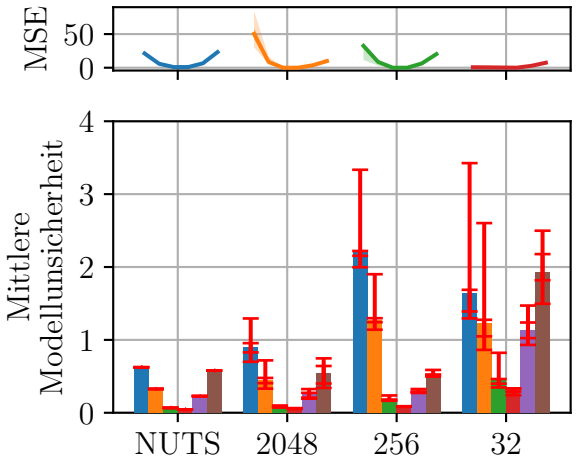
**Abbildung 5.11:** Balkendiagramm der mittleren Modellunsicherheiten für die verschiedenen Daten-Bereiche mit N-Modellen



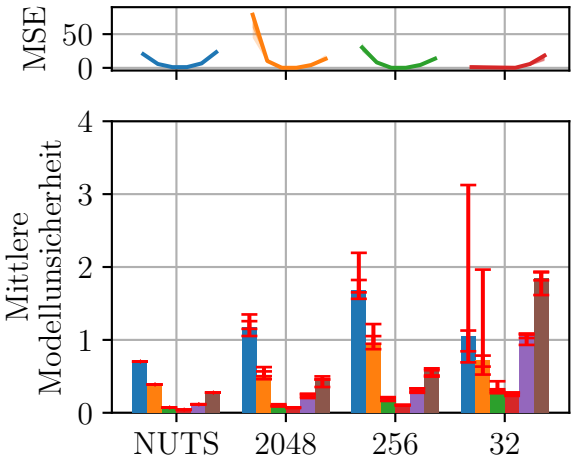

(a) Dropout-Wkt. $p = 0.05$, varianzhomogen



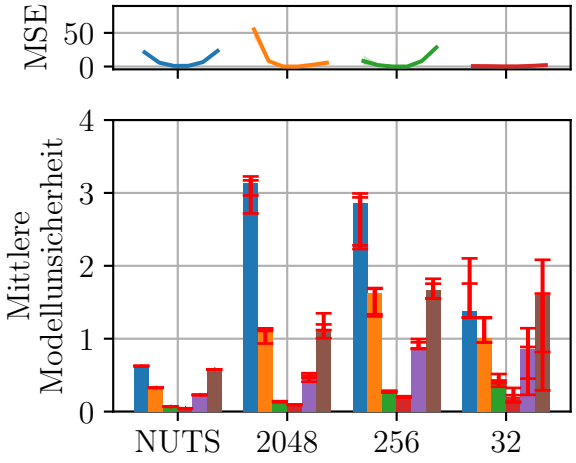

(b) Dropout-Wkt. $p = 0.05$, varianzheterogen



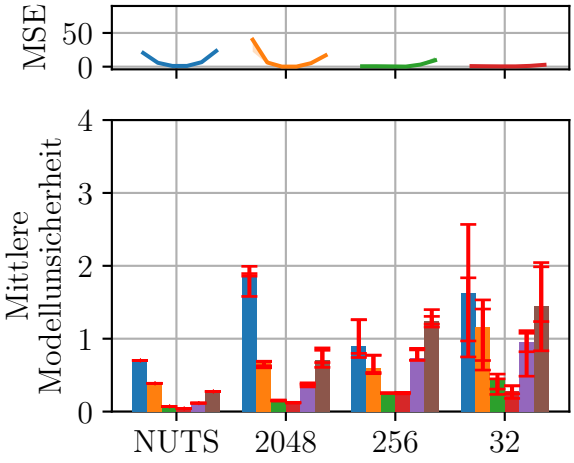

(c) Dropout-Wkt. $p = 0.2$, varianzhomogen




(d) Dropout-Wkt. $p = 0.2$, varianzheterogen




(e) Dropout-Wkt. $p = 0.5$, varianzhomogen




(f) Dropout-Wkt. $p = 0.5$, varianzheterogen

**Abbildung 5.12:** Balkendiagramm der mittleren Modellunsicherheiten für die verschiedenen Daten-Bereiche mit K-Modellen

**K-Modelle** Auch für die K-Modelle scheinen die Modellunsicherheiten das gewünschte Verhalten für bekannte und unbekannte Bereiche zu zeigen. Generell scheinen die Modellunsicherheiten für diesen Modelltyp allerdings größer zu sein als für die N-Modelle. Dies kann daran liegen, dass für die K-Modelle generell größere Modelle für die gleiche Qualität der Funktionsapproximation nötig sind. Da sich gezeigt hat, dass die Modellunsicherheit in einem gewissen Bereich für kleinere Modelle ansteigt, erklärt dies die erhöhte Unsicherheit im Vergleich zu den N-Modellen.
Auch die Tatsache, dass es sowohl für die Modelle mit 32 Neuronen, als auch für die mit 256 Neuronen Fälle gibt, für die die unbekannten Bereiche, welche an den Trainingsbereich mit hoher Datendichte angrenzen (H_außen und H_weit_außen), eine höhere Modellunsicherheit haben als die anderen beiden unbekannten Bereiche. Wie zuvor beschrieben sollte dies erwartungsgemäß nicht der Fall sein. Allerdings war dies für die N-Modelle auch nur für kleine Modelle der Fall. Die Tatsache, dass das Verhältnis der Unsicherheiten für K-Modelle mit 2048 Neuronen wie erwartet ist, spricht wieder dafür, dass nur für ausreichend große Modelle dieses Verhältnis den Erwartungen entspricht.
Durch die Ähnlichkeit der Abbildungen für N-Modelle und K-Modelle, zeigt sich, dass eine Schätzung der Datenunsicherheit anhand des Verfahrens nach [Kendall und Gal, 2017] keinen negativen Einfluss auf die Schätzung der Modellunsicherheit nach [Gal und Ghahramani, 2016b] hat.

**ROC-Kurven** Um die Qualität der Klassifikation von unbekannten Daten anhand der Modellunsicherheit besser bewerten zu können, wurden zusätzlich ROC-Kurven betrachtet. Dabei ergibt sich die Kurve durch das Verändern des Schwellwertes, der angibt ab welchem Wert für die Modellunsicherheit ein Punkt als unbekannt gilt. Für die Bestimmung der ROC-Kurven wurden alle unbekannten Testdaten der Bereiche G_weit_außen, G_außen, H_außen und H_weit_außen verwendet, während aus den bekannten Bereichen G_innen und H_innen zufällige Punkte gewählt wurden, sodass eine ausgeglichene Anzahl an Datenpunkten für die Klassen „bekannt“ und „unbekannt“ entsteht.
Die Abbildung 5.13 zeigt beispielhaft die ROC-Kurven der K-Modelle auf varianzheterogenen Daten und einer Dropout-Rate von $p = 0.5$. Die Kurven der restlichen Modelle zeigen dabei einen ähnlichen Verlauf. Wie gut zu erkennen ist, verlaufen alle Kurven im Bereich der oberen linken Ecke. Daraus folgt, dass alle Modelle grundsätzlich eine

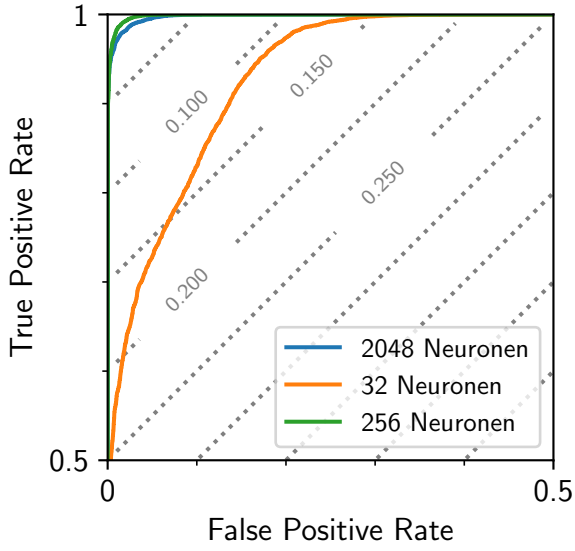
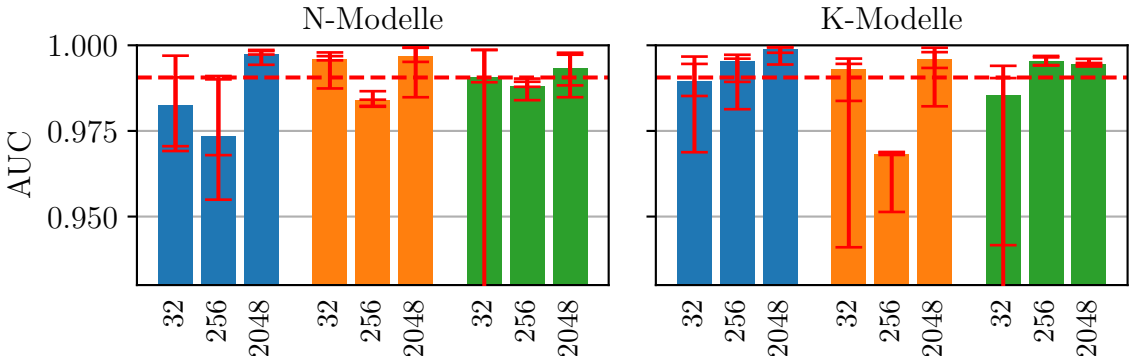
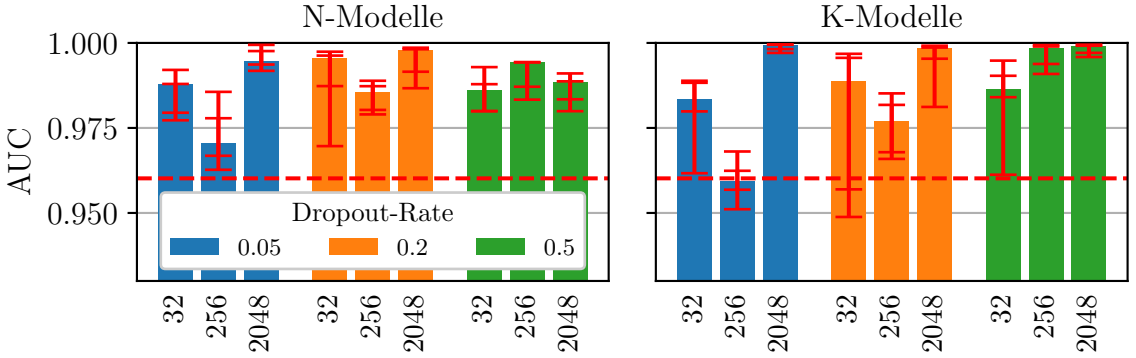



**Abbildung 5.13:** ROC-Kurven für K-Modelle auf varianzheterogenen Daten und Dropout-Rate 0.5.
*Die Abbildung zeigt die ROC-Kurven für drei verschiedenen Modellgrößen von K-Modellen auf varianzheterogenen Daten mit Dropout-Rate* 0.5*. Dabei sind nicht die kompletten Kurven abgebildet, sondern es wurde der obere linke Teil der ROC-Kurven vergrößert.*

gute Klassifikationsleistung erreichen.
Um nun die einzelnen Modelle besser bewerten zu können, wird sich auf die durch die ROC-Kurven eingeschlossene Fläche (area under curve (AUC)) fokussiert. Die Abbildung 5.14 zeigt den AUC-Score in Abhängigkeit zur Dropout-Rate und der Modellgröße für N-Modelle (links) und K-Modelle (rechts) und auf varianzhomogenen (oben) und varianzheterogenen Daten (unten).
Die größten Schwankungen treten erwartungsgemäß bei den Modellen mit 32 Neuronen auf. Für die als ausreichend befundene Modellgröße von 256 Neuronen ergibt die größte Dropout-Rate von $p = 0.5$ die besten Ergebnisse. Auffällig ist jedoch der Einbruch bei den K-Modellen auf varianzhomogenen Daten (oben rechts) für eine Dropout-Rate von $p = 0.2$.
Eine weitere wichtige Beobachtung ist, dass in einigen Fällen die Modelle mit 32 Neuronen einen höheren AUC-Score erreichen als die Modelle mit 256 Neuronen. Daraus kann abgeleitet werden, dass die eigentliche Qualität der Funktionsapproximation nicht




**(a)** AUC-Scores auf varianzhomogenen Daten




**(b)** AUC-Scores auf varianzheterogenen Daten

**Abbildung 5.14:** Balkendiagramm der AUC-Scores in Abhängigkeit der Dropout-Rate

*Die Abbildung zeigt die AUC-Scores der verschiedenen Modelle mit verschiedenen Dropout-Raten und auf verschiedenen Datensätzen. Die verschiedenen Quantile über den verschieden Durchläufen ist durch die roten Fehlerbalken gekennzeichnet. Die rote gestrichelte Linie beschreibt jeweils den AUC-Score der NUTS-Modelle. An der x-Achse sind die jeweiligen Modellgrößen abgetragen während die Dropout-Rate farblich gekennzeichnet ist. Die linken Abbildungen zeigen die Ergebnisse der N-Modelle und rechts sind die der K-Modelle zu sehen.*

entscheidend für das Verhalten der Modellunsicherheit auf unbekannten Daten ist. Anhand der vorherigen Abbildungen hat die Modellkomplexität also nur Auswirkungen auf die Modellunsicherheit im Bereich der Trainingsdaten. Im Vergleich zum NUTS sind die AUC-Scores für den varianzheterogenen Datensatz deutlich geringer als für die Ergebnisse des MC-Dropouts. Auf den varianzhomogenen Daten sind die Ergebnisse wiederum vergleichbar.

#### Fazit

In diesem Abschnitt wurden die Auswirkungen von Modellgröße und Dropout-Rate auf die Modellunsicherheit untersucht. Dabei hat sich ergeben, dass das Verhalten der Modellunsicherheiten nach dem MC-Dropout-Verfahren durchaus Ähnlichkeiten zu dem des NUTS aufweist. Zudem konnten unbekannte Daten anhand der Modellunsicherheit sehr gut erkannt werden. Die Erkennungsleistung ist dabei weitestgehend unabhängig von Modellgröße und Dropout-Rate.

### 5.1.4 Datenunsicherheit

In diesem Abschnitt werden die Datenunsicherheiten, welche durch das Verfahren nach [Kendall und Gal, 2017] geschätzt wurden, untersucht und bewertet. Da für die synthetischen Daten der genaue Rauschterm und dessen Parameter bekannt sind, kann hier der Fehler der Schätzung zu der eigentlichen Datenunsicherheit berechnet werden. So wird im Folgenden, neben den Kalibrierungskurven, der MSE der Schätzung der Datenunsicherheit zur Bewertung herangezogen.
Grundsätzlich ergeben sich dabei einige Erwartungen aus der Tatsache, dass die Schätzung der Datenunsicherheit durch zusätzliche Ausgabeneuronen realisiert wird. Da diese Schätzung demnach einer Approximation einer Funktion gleicht, sollten sich Parameter wie Dropout-Rate und Modellgröße hier ähnlich auswirken, wie auf die Schätzung der Kosinusfunktion. Einziger Unterschied hierbei ist, dass die Approximation der Datenunsicherheit nicht durch überwachtes Training erreicht wird, da diese Funktion in der Regel nicht exakt angegeben werden kann.
Zudem werden in diesem Abschnitt nur noch die K-Modelle betrachtet, da eine Bewertung für die N-Modelle entfällt, da diese keine Datenunsicherheit schätzen.

#### Auswirkungen der Modellgröße

Für die Untersuchung der Auswirkung der Modellgröße auf die Datenunsicherheit wurden die zuvor trainierten K-Modelle, anhand derer in Abschnitt 5.1.1 (Seite 61) eine geeignete Modellgröße gesucht wurde, verwendet. Das bedeutet es wurden Modelle betrachtet, welche mit einer Dropout-Rate von $p = 0.5$ auf den varianzhomogenen Daten trainiert wurden.
Die Abbildung 5.15 zeigt nun den Verlauf der mittleren Datenunsicherheit der K-

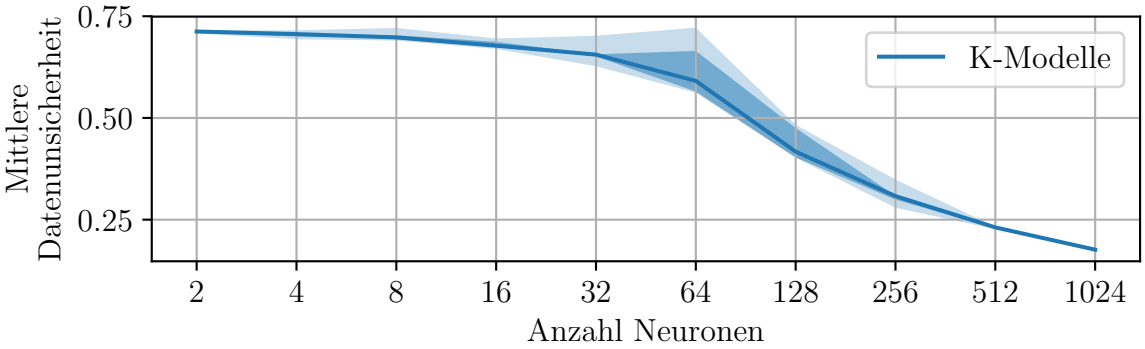



**Abbildung 5.15:** Mittlere Datenunsicherheit in Abhängigkeit der Modellgröße.
*Die Abbildung zeigt den Verlauf der mittleren Datenunsicherheit in Abhängigkeit der Modellgröße.*

Modelle. Wie erwartet sinkt die geschätzte Datenunsicherheit mit der Größe der Modelle. Allerdings ist auffällig, dass Modelle mit 64 und 128 Neuronen eine erhöhte Varianz zwischen verschiedenen Initialisierungen haben als alle anderen Modelle.
Eine mögliche Erklärung hierfür ist, dass für diese Modellgrößen ausgehend von einigen Initialisierungen ein Modell seinen Trainingsfehler (negative Log-Likelihood) durch eine komplexere Datenunsicherheit verringert, anstatt die eigentlichen Daten besser abzubilden. Wie durch den Median der Kurven und der darunter liegenden Fläche erkennbar ist, folgt der Großteil der Modelle dem erwarteten Verlauf. Diese Modelle könnten also ihren Fehler durch eine bessere Abbildung der Kosinusfunktion minimiert haben, während bei den Modellen mit einer größeren mittleren Datenunsicherheit der oben beschriebene Fall eingetreten ist.

### MSE der Schätzung

Um einen Überblick darüber zu bekommen, wie sich der Fehler der Schätzung der Datenunsicherheit mit steigender Modellkomplexität verhält wird zuerst der Verlauf des Fehlers in Abhängigkeit der Modellgröße betrachtet. Die Abbildung 5.16 zeigt hierfür den Fehler als Funktion der Modellgröße. Für die y-Achse wurde dabei eine symmetrische logarithmische Skala[14] verwendet, da der Ausschlag bei Modellen mit 64 Neuronen ansonsten die Abbildung zu sehr dominiert hätte. Erkennbar ist, dass

[14] Logarithmische Skala, wobei für einen Bereich um den Nullpunkt eine linear Skala verwendet wird. Dies hat den Vorteil, dass der Nullpunkt abgetragen werden kann.

generell der Fehler mit steigender Modellgröße abnimmt. Dies entspricht der Erwartung, da die Schätzung der Datenunsicherheit der Approximation einer Funktion gleicht. Je größer das Modell, desto besser kann dieses die eigentliche Funktion der Datenunsicherheit abbilden. Zudem ist der Fehler der Datenunsicherheit abhängig vom Fehler auf der Kosinusfunktion, da ein Modell diese zuerst ausreichend abbilden muss, bevor die richtige Funktion für die Datenunsicherheit gelernt werden kann. Damit ist der Fehler bei der Schätzung der Datenunsicherheit für kleiner Modelle größer, da diese zu klein sind, um die Kosinusfunktion abzubilden.
Allerdings ist auch hier ein auffälliges Verhalten für die Modellgrößen um 64 Neuronen zu beobachten. In diesem Bereich zeigt sich, neben einem Sprung im Fehler, eine starke Varianz über den fünf Durchläufen. Auch im Verlauf der Modellunsicherheit war zuvor eine Änderung an dieser Stelle zu erkennen.
Erst am Fehler ist ersichtlich, dass der Verlauf der mittleren Datenunsicherheit, wie zuvor angenommen, im Mittel nicht dem erwarteten Verlauf folgt. Es ist deutlich, dass der Fehler der Schätzung der Datenunsicherheit für das Modell mit 64 Neuronen deutlich von dem Wert abweicht, den man auf Basis der restlichen Kurve erwarten würde. Auffällig ist dabei, dass der Fehler für Modelle mit 64 Neuronen größer ist als für Modelle mit 2 Neuronen. Für derart kleine Modelle kann angenommen werden, dass diese den Mittelwert der Zielwerte lernen und jegliche Abweichung von diesem durch Datenunsicherheit erklären. Das bedeutet, die dabei geschätzte Datenunsicherheit sollte

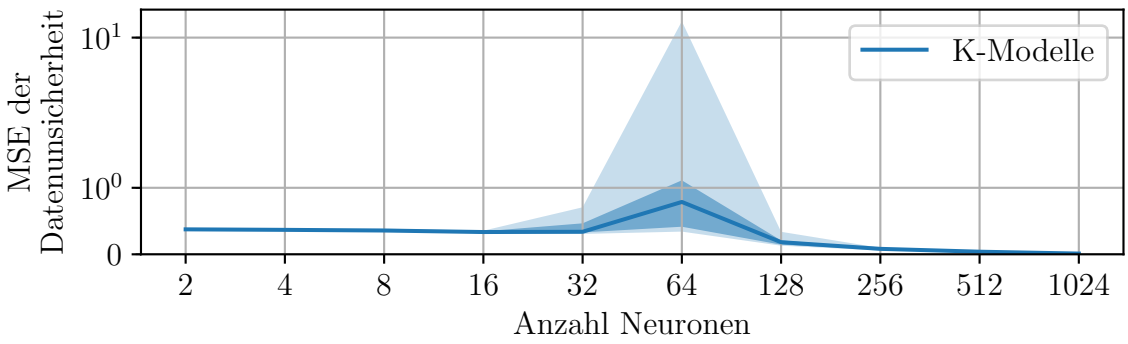



**Abbildung 5.16:** Verlauf des MSE der Datenunsicherheit in Abhängigkeit der Modellgröße

*Die Abbildung zeigt den Verlauf des MSE zwischen der geschätzten Datenunsicherheit und den exakten Parametern des Rauschterms als Funktion der Modellgröße.*

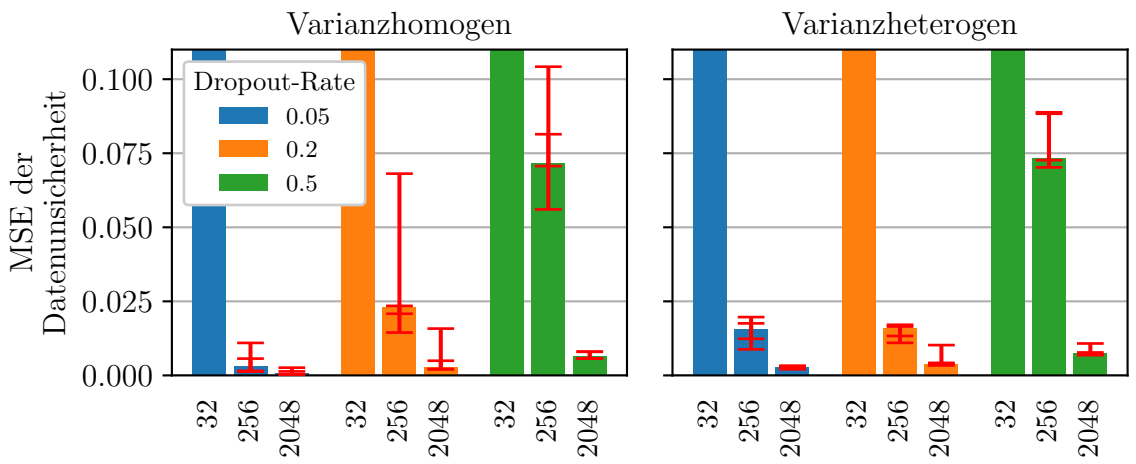



**Abbildung 5.17:** MSE der Schätzung der Datenunsicherheit.
*Die Abbildung zeigt den MSE der Schätzung der Datenunsicherheit für K-Modelle mit verschiedenen Dropout-Raten und auf verschiedenen Datensätzen. An der x-Achse sind dabei die Modellgrößen abgetragen, während die verschiedenen Dropout-Raten farblich markiert sind.*

eine obere Grenze darstellen, da der Mittelwert der Zielwerte die schlechteste Schätzung darstellt. Da allerdings der Fehler der Modelle mit 64 Neuronen größer ist und die Datenunsicherheit eine rein positive Zahl ist, bedeutet dies, dass die Schätzung dort teilweise größer sein muss. Möglicherweise wird bei diesen Modellen zwar eine bessere Abbildung der Kosinusfunktion gelernt, allerdings auf Kosten der Datenunsicherheit. Falls die gelernte Funktion der Datenunsicherheit keiner Konstanten gleich sondern eine gewisse Steigung besitzt, so wären die geschätzten Datenunsicherheiten an manchen Stellen sehr groß und würden somit zu einem extremen Fehler im Bezug auf den eigentlichen Rauschterm ergeben.
Abbildung 5.17 zeigt nun einen Vergleich der Fehler der Schätzung der Datenunsicherheit für verschiedene Dropout-Raten und für die verschiedenen Datensätze. Leicht zu erkennen ist, dass die Modelle mit 32 Neuronen einen größeren Fehler als die restlichen Modelle machen. In der Abbildung wurden die Fehler dieser Modelle abgeschnitten, um die restlichen Ergebnisse besser untersuchen zu können. Der große Fehler dieser Modelle ist, wie zuvor erwähnt, leicht erklärbar, da diese schon zu klein sind, um den Kosinus abbilden zu können. Da sie nun zusätzlich die Funktion der Datenunsicherheit abbilden müssen, fallen die Fehler dementsprechend groß aus.

Für die restlichen Modelle können zwei Zusammenhänge erkannt werden. Je größer das Modell, desto besser kann die Datenunsicherheit geschätzt werden und je größer die Dropout-Rate, desto größer ist der Fehler der geschätzten Datenunsicherheit. Dieses Beobachtungen sind wieder analog zu dem Verhalten der Schätzung einer vorgegebenen Funktion, wie dem Kosinus, da, wie zuvor beschrieben, die Schätzung der Datenunsicherheit der Abbildung einer Funktion gleicht.

### Kalibrierungskurven

Neben der Berechnung des Fehlers zwischen Schätzung und Ground Truth kann für die Datenunsicherheit zudem die Kalibrierungskurve für eine Bewertung herangezogen werden. Im Gegensatz zu der Modellunsicherheit sollte die Datenunsicherheit die Fehler des Modells ausreichend gut erklären. Die Kalibrierungskurve kann dabei über dem

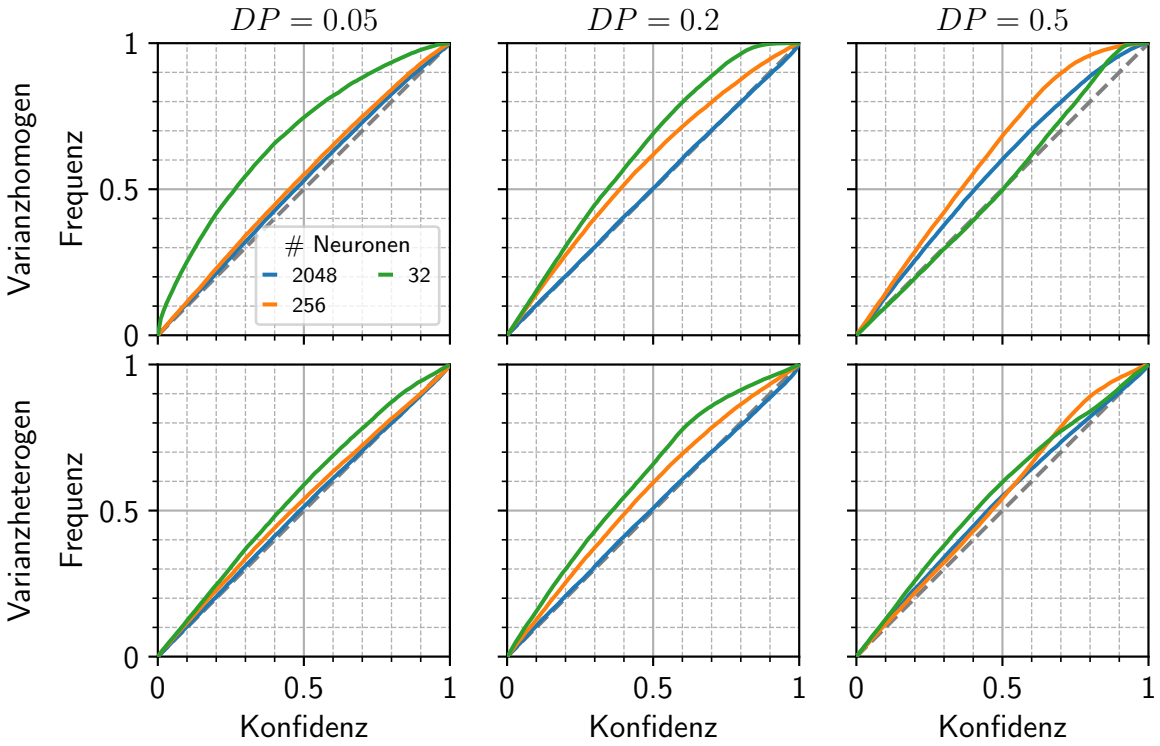



**Abbildung 5.18:** Kalibrierungskurven der K-Modelle.
*Zu sehen sind die Kalibrierungskurven der K-Modelle für verschiedene Datensätze und Dropout-Raten. Dabei zeigt die obere Zeile die Ergebnisse auf dem varianzhomogenen und die untere auf den varianzheterogenen Daten. Die Spalten zeigen jeweils die Ergebnisse für die verschiedenen Dropout-Raten.*

Testdaten berechnet werden.
Abbildung 5.18 zeigt hierfür die Kalibrierungskurven der K-Modelle für die verschiedenen Dropout-Raten, Modellgrößen und Datensätze. Für die Dropout-Raten $p = 0.05$ und $p = 0.2$ sind die Modelle mit 2048 Neuronen jeweils am besten kalibriert. Das liegt daran, dass, wie zuvor festgestellt, eine kleine Dropout-Rate dazu führt, dass das Modell die Datenunsicherheit besser schätzen kann. Je besser ein Modell die Kosinusfunktion und die Standardabweichung des Rauschterms lernen kann, desto besser ist dessen Kalibrierung.
Allerdings ist auch erkennbar, dass auch die Modelle mit 32 Neuronen, die weder die Kosinusfunktion noch die eigentliche Datenunsicherheit gut abbilden konnten, gut kalibriert sind. Für eine Dropout-Rate von $p = 0.5$ ist die Qualität der Kalibrierung sogar sehr ähnlich und im Falle der varianzhomogenen Daten sogar besser. Dies lässt erkennen, dass eine gute Kalibrierung nicht bedeutet, dass das Modell die Funktion gut approximiert hat, sondern nur, dass es seine Fehler ausreichend durch die Datenunsicherheit erklärt.
Um die Qualität der Kalibrierungskurven und deren Varianz über den Durchläufen besser beurteilen zu können ist in Abbildung 5.19 der Fehler der Kalibrierungskurven (Expected Calibration Error (ECE)) für alle Modelle abgetragen. Neben der Tatsache, dass der Kalibrierungsfehler der Modelle mit 32 Neuronen am stärksten schwankt, ist zu erkennen, dass diese Modelle auf den varianzheterogenen Daten besser kalibriert

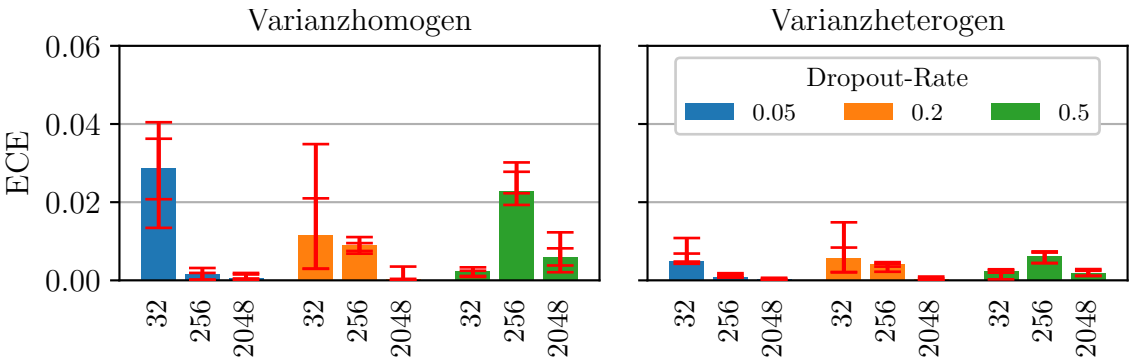



**Abbildung 5.19:** MSE der Kalibrierungskurven.
*Die Abbildung zeigt den Fehler der Kalibrierungskurven gemessen am MSE für die K-Modelle mit verschiedener Dropout-Rate, Modellgröße und auf unterschiedlichen Datensätzen.*

sind als auf den varianzhomogenen Daten. Grundsätzlich scheint dies für alle Modelle und Dropout-Raten der Fall zu sein, obwohl der varianzhomogene Datensatz einen konstanten Rauschterm hat und dieser damit leichter zu approximieren sein sollte.
Auch ist zu sehen, dass die Modelle mit 2048 Neuronen generell die beste Kalibrierung aufweisen. Deutliche Ausnahme hierbei bilden allerdings die Modelle mit Dropout-Rate $p = 0.5$ auf varianzhomogenen Daten. Hier sind die Modelle mit 32 besser kalibriert als die mit mehr Neuronen. Dies könnte durch die Form der Kosinusfunktion und der Wahl des Rauschterms begründet sein, da eine sehr simple Approximation möglicherweise Fehler erzeugt, die leicht durch eine einfache Funktion der Datenunsicherheit erklärt werden können. Das bedeutet die Verteilung dieser Fehler gleicht zu sehr einer Normalverteilung, als dass eine komplexe Funktion für die Datenunsicherheit nötig wäre, um eine gute Kalibrierung zu erreichen.

**Fazit**

In diesem Abschnitt wurde das Verhalten der Datenunsicherheiten, welche nach dem Verfahren nach [Kendall und Gal, 2017] geschätzt wurden, untersucht. Dabei wurde die Abhängigkeit der Datenunsicherheit von der Modellgröße und der Dropout-Rate untersucht. Zudem wurden die Unsicherheiten anhand des MSE und der Kalibrierung bewertet. Dabei hat sich gezeigt, dass die eine falsche Schätzung der Datenunsicherheit nicht bedeutet, dass diese nicht kalibriert ist. Im Allgemeinen wiesen die geschätzten Datenunsicherheiten eine sehr gute Kalibrierung auf.

## 5.2 STEAG Prozessdaten

Nachdem im bisherigen Teil der Experimente eine Basis für die Einschätzung der beiden Unsicherheiten und der verwendeten Verfahren zur Schätzung dieser erarbeitet wurde, werden im Folgenden die Untersuchungen auf den Prozessdaten der SES beschrieben. Der Fokus liegt dabei, wie im Abschnitt 4.2 (Seite 50) beschrieben, auf der Erkennung der Anomalien im Datensatz.
Neben Experimenten zu der Anomaliedetektion werden zudem vorgenommene Untersuchungen zur Modellfindung und allgemeinen Betrachtungen der Unsicherheiten aufgeführt. Für alle Experimente wurden dabei die gleichen Hyperparameter verwendet, wie bei den synthetischen Daten. Tabelle 5.2 listet hierfür nochmals alle fest gewählten

| **Hyperparameter** | **Wert** |
|---|---|
| Aktivierung | ReLU ([Nair und Hinton, 2010]) |
| Batch-Größe | 200 |
| Optimierer | AdaDelta ([Zeiler, 2012]) |
| Max. Epochen | 5000 |
| Dropout-Rate | 0.5 |
| Early-Stopping[15] | 2000 |

**Tabelle 5.2:** Tabelle der verwendeten Hyperparameter beim Training auf Daten der SES.
*Die hier dargestellte Tabelle beschreibt die verwendeten Hyperparameter beim Training aller Modelle, welche auf den Daten der SESangewandt wurden.*

Hyperparameter auf. Im Unterschied zu den Modellen für die synthetischen Daten wurde für die folgenden Experimente eine Dropout-Rate von $p = 0.5$ festgesetzt. Aus den Untersuchungen auf den synthetischen Daten ist abzuleiten, dass eine geringere Dropout-Rate sich zwar positiv auf die Modellgüte auswirkt, allerdings nicht auf die Kalibrierung der Datenunsicherheit oder das Verhalten der Modellunsicherheit für unbekannte Daten. Daher kann sich auf eine Dropout-Rate festgelegt werden, um sich bei den Experimenten auf die Klassifizierung der Anomalien konzentrieren zu können. Generell ist zudem anzumerken, dass, wie bereits in Abschnitt 4.2.1 beschrieben, der MSE für die folgenden Modelle mit der Varianz der Daten skaliert wird. Allerdings wird die Bezeichnung durch „MSE“ als Vereinfachung beibehalten.

### 5.2.1 Modellarchitektur

Bevor eine Detektion der Anomalien anhand von Unsicherheiten getestet werden kann muss, auch für diesen Datensatz erst eine angemessene Modellarchitektur gefunden werden. Da sich die Datensätze für die beiden beschriebenen Herangehensweisen unterscheiden, gilt es für beide eine geeignete Architektur zu finden. Das Modell, welches für die Anomaliedetektion mittels der Datenunsicherheit verwendet wird, bildet simultan alle Eingaben auf alle Ausgabekanäle ab. Damit ist zu erwarten, dass dieses Modell

[15] Falls sich nach 2000 Epochen der Fehler auf den Validierungsdaten nicht verbessert hat wird das Training beendet.

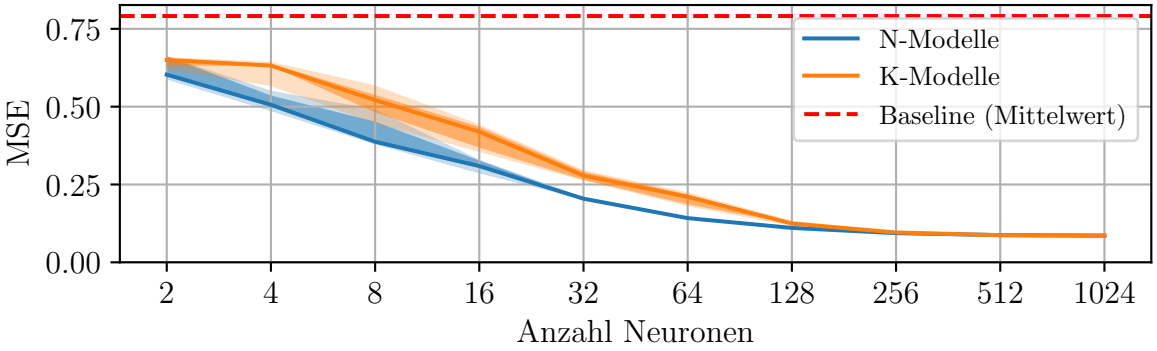



**Abbildung 5.20:** Verlauf des MSE in Abhängigkeit der Modellgröße
*Die Abbildung zeigt den MSE als Funktion der Modellgröße für die N-Modelle und K-Modelle. Die rote gestrichelte Linie beschreibt die Mittelwert-Baseline* $y_{Mittelwert}$.

größer sein muss, als die Modelle, welche alle Eingaben auf nur einen Ausgabekanal abbilden müssen. Deshalb wird zuerst eine geeignete Architektur für die Herangehensweise basierend auf der Datenunsicherheit gesucht, um anschließend kleinere Modelle für den Ansatz auf Basis der Modellunsicherheit zu finden.
Da die Modelle anhand derer die Labelinformationen generiert wurden nur eine Hidden-Schicht und nicht mehr als 10 Neuronen besaßen, wurde sich für potentielle Modellarchitekturen analog zu denen, welche für die synthetischen Daten in Abschnitt 5.1.1 (Seite 61) getestet wurden, entschieden. Zudem wurde die Mittelwert-Baseline[16] $y_{Mittelwert}$, analog zu den synthetischen Daten, definiert, für die die Ausgabe konstant dem Mittelwert der Zielwerte in den Trainingsdaten gleicht.
Die Abbildung 5.20 zeigt nun den Fehler für die verschiedenen Modelle. Gut erkennbar ist, dass die K-Modelle etwas größer sein müssen, um vergleichbare Werte zu den N-Modellen zu erreichen. Dies sollte besonders für reale Daten zutreffen, da hier anzunehmen ist, dass Rauschterme eine komplexere Form besitzen, als dies bei den synthetischen Daten der Fall war. Diese komplexeren Rauschterme würden damit mehr Parameter des Modells beanspruchen, damit diese ausreichend abgebildet werden können.
Anhand der Abbildung ist zudem erkennbar, dass der Fehler ab einer Modellgröße von 256 Neuronen für beide Modelltypen konvergiert. Damit ist anzunehmen, dass 256 Neuronen ausreichend für diesen Datensatz sind. Allerdings werden für die Experimente

[16] Für die mathematische Definition sei auf 5.1 verwiesen, wobei der Fehler anhand der Varianz der jeweiligen Ausgabekanäle skaliert wurde.

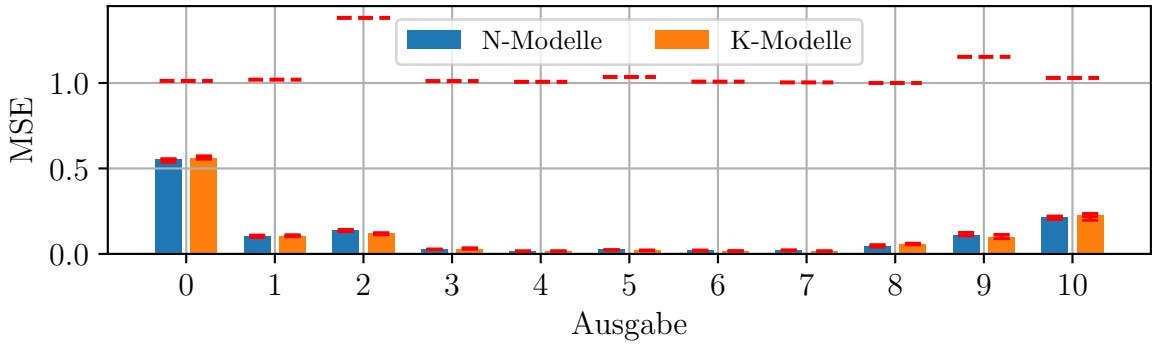



**Abbildung 5.21:** MSE pro Ausgabe
*Zu sehen sind die MSE-Werte der Modelle mit* 1024 *Neuronen für jede Ausgabe einzeln. Die roten gestrichelten Linien beschreiben die jeweilige Mittelwert-Baseline der Ausgaben.*

dennoch 1024 Neuronen pro Schicht verwendet, da die Ergebnisse auf den synthetischen Daten gezeigt haben, dass eine Überschätzung der Modellgröße keine negativen Auswirkungen auf die Datenunsicherheit bzw. Erkennbarkeit von unbekannten Daten anhand der Modellunsicherheit hat, besonders bei einer Dropout-Rate von $p = 0.5$. Zusätzlich werden Modelle mit 32 Neuronen im Kontext der Modellunsicherheit betrachtet, um festzustellen, ob auch für reale Daten gilt, dass der Fehler des Modells einen geringen Einfluss auf die Erkennbarkeit von unbekannten Daten hat.

### MSE pro Ausgabe

Da die Modelle für den Datensatz der SES mehrere Ausgaben haben, ist es von Interesse festzustellen, wie sich der Fehler dieser Ausgaben im Einzelnen verhält. Hierfür zeigt die Abbildung 5.21 die Fehler der Ausgaben für Modelle mit 1024 Neuronen. Zu sehen ist, dass für den Großteil der Ausgaben ein geringer Fehler erzielt werden kann. Allerdings stellen die Ergebnisse auf der Ausgabe 0 eine auffällige Ausnahme dar. Hier ist der Fehler beider Modelltypen um ein vielfaches größer, als für die übrigen Ausgaben.
Ein bedeutender Unterschied der Fehler zwischen den Modelltypen ist nicht zu erkennen. Das bedeutet, dass die K-Modelle auf einem realen Datensatz die gleiche Qualität der Schätzung der Zielwerte erreichen wie die N-Modelle. Allerdings bieten die K-Modelle zusätzlich eine Information über Datenunsicherheit, was in vielen Anwendungen vorteilhaft sein kann. Da K-Modelle kaum einen Nachteil mit sich bringen, kann die

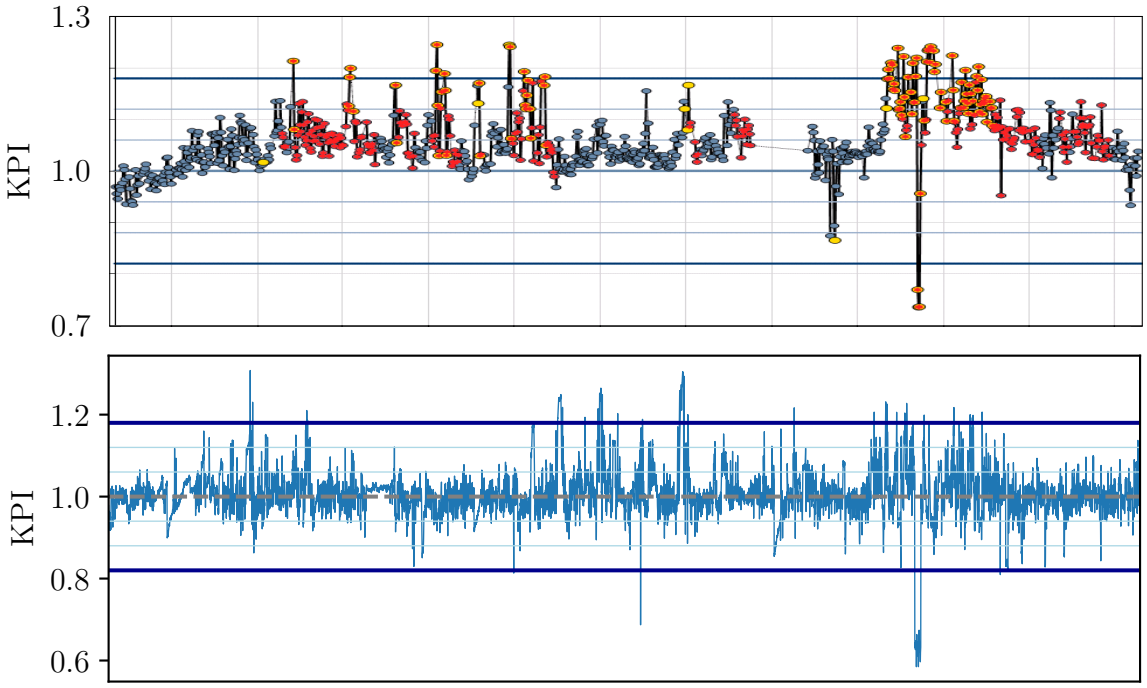
Verwendung dieser für die Anwendung auf den Prozessdaten der SES empfohlen werden.

### Vergleich mit Modellen der SES

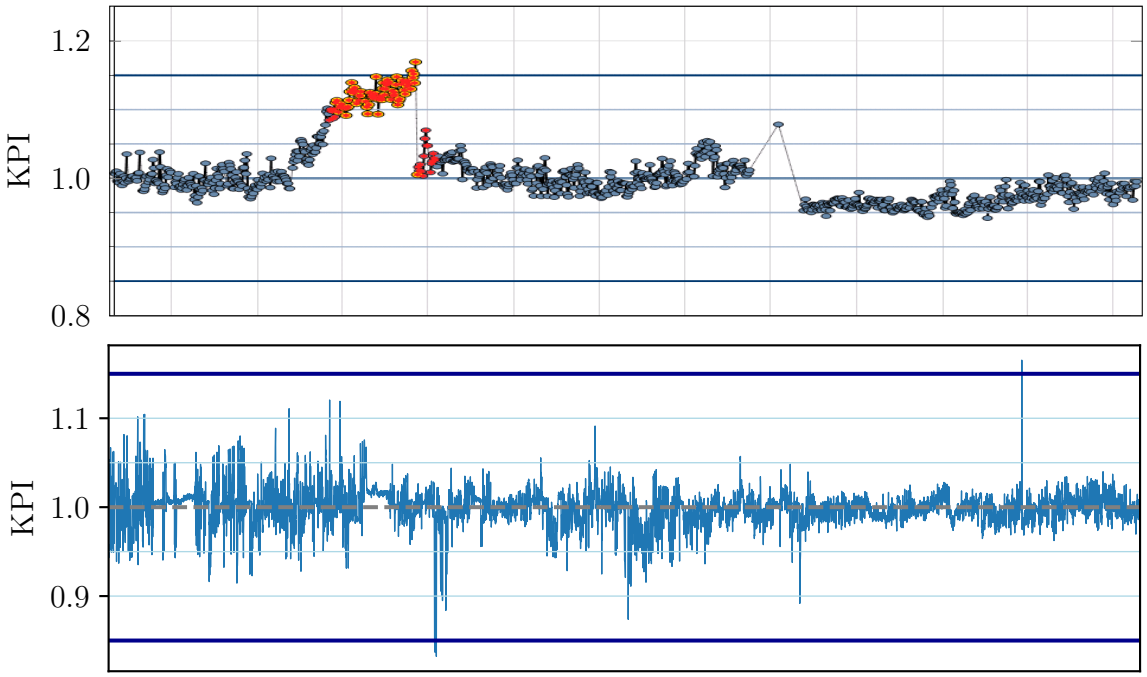
Um abschätzen zu können, ob die in dieser Arbeit trainierten Modelle eine ausreichende Qualität haben, werden diese mit den Modellen der SES, welche zur Erstellung der Labeldaten verwendet wurden, verglichen. Allerdings liegen für die Modelle der SES keine exakten Werte der Modellqualität vor, da die Details der Experimente nicht mehr verfügbar sind und eine erneute Durchführung zu viel Zeit beanspruchen würde. Lediglich die ungefähren Werte der für die Anomaliedetektion berechneten KPI-Werte (key performance indicator) sind bekannt. Dies ist allerdings ausreichend, um festzustellen, ob die in dieser Masterarbeit trainierten Modelle eine vergleichbare Qualität zu denen der SES besitzen.
Der KPI-Wert für den i-ten Datenpunkt wurde dabei nach folgender Formel berechnet:

$$KPI_i = \frac{t_i - y_i}{C} + 1 \tag{5.3}$$

Dabei steht $C$ für eine Konstante, welche für jeden der vorhandenen Ausgabekanäle fest definiert ist und von der SES vorgegeben wurden. Anhand der Formel ist zu erkennen, dass diese KPI-Werte einen skalierten und verschobenen Fehler zwischen den Zielwerten $t_i$ und den Ausgaben eines Modells $y_i$ darstellen. Damit kann anhand dieser Werte abgeschätzt werden, ob die in dieser Arbeit trainierten Modelle qualitativ von den Modellen der SES abweichen. Im Falle eines guten Modells sollten sich die KPI-Werte um den Wert eins bewegen.
Zudem wurden für die Anomaliedetektion Kontrollgrenzen für die Ausgabekanäle definiert. Diese Grenzwerte dienen allerdings in dieser Arbeit nur, um den visuellen Vergleich der KPI-Werte zu erleichtern. Deshalb wird das Vorgehen bei der Bestimmung oder Verwendung dieser Grenzwerte nicht weiter eingegangen.
Um die Ergebnisse des Vergleichs übersichtlich zu gestalten werden hier nur ausgewählte Ausgabekanäle verglichen. Die Gegenüberstellung aller verfügbaren KPI-Verläufe kann allerdings im Anhang A.2.1 eingesehen werden.
Abbildung 5.22 zeigt nun den Verlauf der KPI-Werte für die Modelle der SES und für das N-Modell mit 1024 Neuronen für Ausgabekanal 0 (5.22a) beziehungsweise 1 (5.22b).




**(a)** Vergleich für Ausgabekanal 0




**(b)** Vergleich für Ausgabekanal 1

**Abbildung 5.22:** Vergleich der KPI-Werte für Ausgabekanal 0 und 1.
*Vergleich der Verläufe der KPI-Werte für Modelle der SES (jeweils obere Kurve) und Modell dieser Arbeit (jeweils untere Kurve).*

Dabei zeigt die Kurve der Modelle der SES für Ausgabekanal 0 eines der schlechtesten Ergebnisse und für Ausgabekanal 1 eines der Besten. Die farbigen Markierungen in den jeweils oberen Grafiken können dabei ignoriert werden, da diese die Anomalielabel darstellt[17]. Wichtig ist für den Vergleich der Kurven, dass sich der Datensatz zwischen den Modellen unterscheidet. Zwar kommen die Daten der beiden Datensätze aus dem gleichen Zeitraum, allerdings sind beide Datensätze nicht identisch. Deshalb können Ähnlichkeiten der Kurven beobachtet werden, wie beispielsweise der Ausschlag in Abbildung 5.22a im letzten Drittel. Der größte Unterschied der Modelle liegt allerdings in der Tatsache, dass die Modelle der SES nicht auf Anomalien trainiert wurden.
Anhand der Kurven ist erkennbar, dass das Modell dieser Arbeit zwar allgemein etwas schlechtere KPI-Werte verursacht, diese allerdings nicht allzu stark von denen der Modelle der SES abweichen. Das bedeutet, es kann angenommen werden, dass die Qualität der Modelle vergleichbar ist und das trainierte Modell eine ausreichende Qualität besitzt.

#### Modelle für Anomaliedetektion anhand der Modellunsicherheit

Da ein geeignetes Modell für die Anomaliedetektion basierend auf der Datenunsicherheit gefunden wurde und dieses zudem vergleichbare Ergebnisse zu den Modellen der SES liefert, können die Modelle für den zweiten Ansatz der Anomaliedetektion untersucht werden. Dabei können die Modelle, welche nur einen der Ausgabekanäle abbilden müssen, jeweils mit dem Ergebnis des großen Modells für den jeweiligen Kanal verglichen werden. Damit ist kein zweiter Vergleich zu den Modellen der SES nötig.
Da die Modelle für die Anomaliedetektion anhand der Modellunsicherheit vermutlich kleiner sein können, wurden hier zunächst nur Modellgrößen von 2 bis 128 Neuronen getestet. Abbildung 5.23 zeigt den Verlauf des Fehlers für jeden Ausgabekanal. Dabei sind neben der Mittelwert-Baseline zusätzlich die Fehler des besten Modells mit 1024 Neuronen aus den vorherigen Untersuchungen abgetragen.
Zu erkennen ist, dass der Verlauf des Fehlers ab einer Größe von 32 Neuronen abflacht. Zudem sind manche Modelle im Vergleich zu dem großen Modell mit 1024 Neuronen etwas schlechter. Allerdings erreichen die meisten Modelle einen vergleichbaren Fehler.

[17] Für den Vergleich der Modelle standen nur die fertigen Abbildungen der SES zur Verfügung und nicht die Rohdaten.

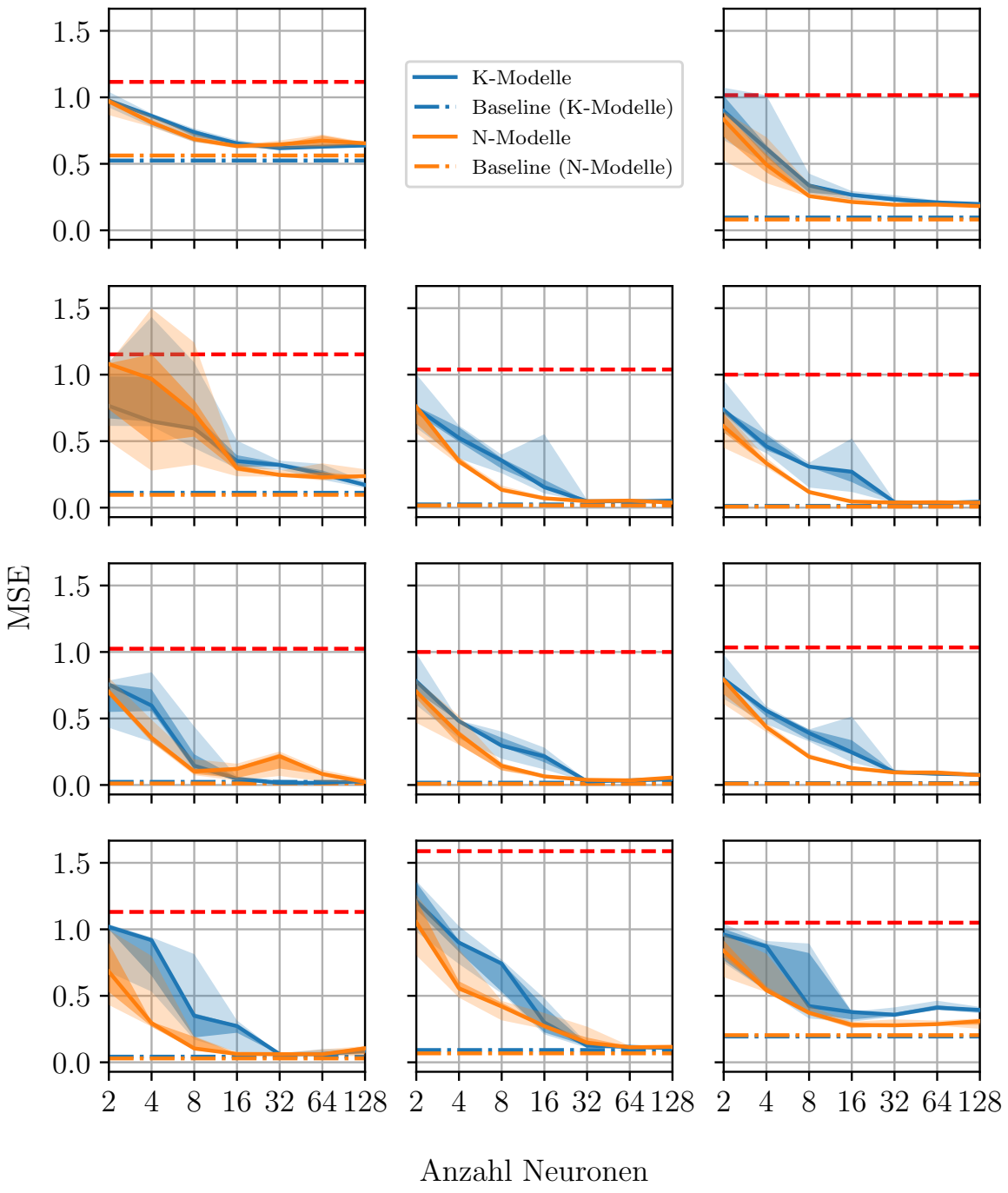



**Abbildung 5.23:** Verlauf des Fehlers für Anomaliedetektion anhand der Modellunsicherheit

*Zu sehen ist jeweils der Verlauf des Fehlers für jeden Ausgabekanal und für beide Modelltypen.*

Die Ausgabekanäle, für die der Unterschied zwischen den Modellen der beiden Herangehensweisen am größten ist, sind zudem genau die Kanäle, auf denen das Modell für die Anomaliedetektion anhand der Datenunsicherheit die größten Fehler gemacht hat[18]. Das bedeutet, dass generell angenommen werden kann, dass die verschiedenen Modelle ähnliche Funktionen für die Ausgabekanäle lernen, da deren Fehler vergleichbar sind. Anzumerken ist dabei, dass die Experimente auf den synthetischen Daten ergeben haben, dass der Fehler des Modells keine großen Auswirkungen auf die Erkennbarkeit von unbekannten Daten hat. Daher kann ein leicht höherer Wert für den MSE der hier trainierten Modelle in Kauf genommen werden und sollte keine Relevanz für die Qualität der Anomaliedetektion haben.
Anhand der Ergebnisse kann angenommen werden, dass die Modelle mit 128 Neuronen für die Abbildung eines einzigen Ausgabekanals ausreichend sind. Somit werden diese Modelle in den späteren Experimenten zur Erkennung der Anomalien anhand der Modellunsicherheit verwendet.

**Fazit**

In diesem Abschnitt wurden verschiedene Modellgrößen, analog zu den Experimenten auf den synthetischen Daten in Abschnitt 5.1.1 (Seite 61), getestet und es hat sich gezeigt, dass 256 Neuronen mit zwei Hidden-Schichten ausreichend ist, um alle Ausgabekanäle simultan abzubilden. Für die weiteren Experimente werden dennoch 1024 Neuronen verwendet, da sich gezeigt hat, dass eine Überschätzung der Modellgröße keine Nachteile mit sich bringt. Für die Abbildung eines einzigen Ausgabekanals haben sich die Modelle mit 128 Neuronen als ausreichend herausgestellt und werden in den jeweiligen Experimenten zur Erkennung der Anomalien verwendet.
Zudem wurden diese Modelle mit denen der SES verglichen, wobei sich gezeigt hat, dass die Modelle vergleichbare Ergebnisse in Bezug auf den Modellfehler erzielen. Die hier trainierten Modelle schneiden dabei augenscheinlich etwas schlechter ab.

[18] Vergleich Abbildung 5.21 (Seite 94)

### 5.2.2 Allgemeine Untersuchung der Unsicherheiten

Da nun eine geeignete Modellarchitektur gefunden wurde und diese vergleichbare Ergebnisse zu den Modellen der SES erreicht, kann nun eine allgemeine Betrachtung der beiden Unsicherheitstypen vorgenommen werden. Dabei liegt der Fokus im Vergleich zu den Experimenten auf den synthetischen Daten und den dabei gemachten Beobachtungen. Für den Fall, dass diese sich auf den Daten der SES wiederholen, kann eine Abhängigkeit zu dem verwendeten Datensatz ausgeschlossen werden. Es werden dabei die Modelle betrachtet, welche alle elf Ausgabekanäle simultan abbilden müssen, da dies die komplexere Aufgabenstellung darstellt. Somit ist davon auszugehen, dass sich die Modelle für nur einen Ausgabekanal ähnlich verhalten.

#### Modellunsicherheit

Analog zu den Untersuchungen über den synthetischen Daten wird zuerst der Verlauf der Modellunsicherheit in Abhängigkeit der Modellgröße betrachtet. Abbildung 5.24 zeigt besagten Verlauf für die beiden Modelltypen. Auch hier ist zu erkennen, dass für kleinere Modelle (bis 8 Neuronen) die Modellunsicherheit ansteigt und stark durch die Initialisierung der Gewichte beeinflusst wird. Für größere Modelle sinkt sowohl die Modellunsicherheit als auch deren Varianz über den verschiedenen Initialisierungen. Dies bildet den schon in Abschnitt 5.1.3 beobachteten Hügel. Damit ist auszuschließen, dass es sich bei diesem Verhalten um eine Eigenheit des synthetischen Datensatzes handelt, sondern um eine allgemeine Eigenschaft der Modellunsicherheit, welche durch

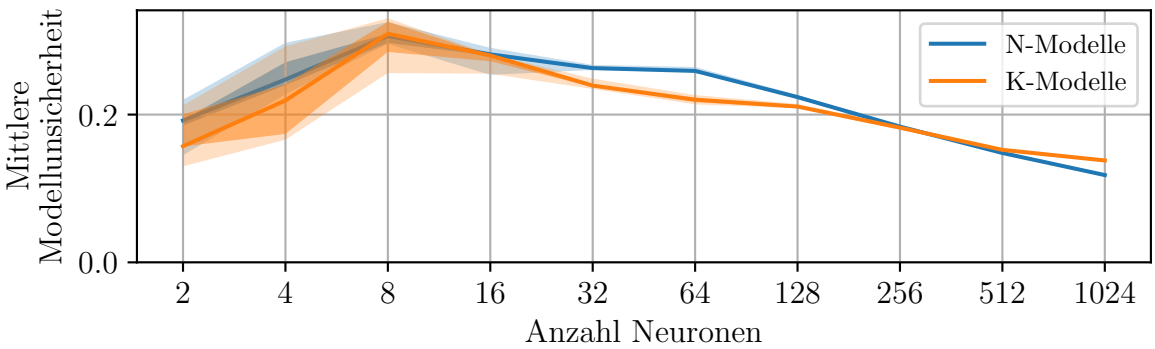



**Abbildung 5.24:** Verlauf der Modellunsicherheit in Abhängigkeit der Modellgröße
*Die Abbildung zeigt den Verlauf der Modellunsicherheiten als Funktion der Modellgröße. Dabei sind die Ergebnisse für N-Modelle und K-Modelle abgetragen.*

MC-Dropout berechnet werden.

### Datenunsicherheit

Abbildung 5.25 zeigt den Verlauf der Datenunsicherheit in Abhängigkeit der Modellgröße. Zu erkennen ist, dass diese mit der Größe der Modelle sinkt und sich ab einer Anzahl von 512 Neuronen pro Schicht kaum ändert. Dies legt nahe, dass ab dieser Modellgröße das Modell sowohl die Ausgabekänale als auch deren Datenunsicherheiten abbilden kann.
Die für die synthetischen Daten beobachtete erhöhte Varianz der mittleren Datenunsicherheit über den verschiedenen Durchläufen ist hier nicht in vergleichbarem Ausmaß zu beobachten. Es ist allerdings eine leicht erhöhte Varianz für Modelle mit 8 beziehungsweise 16 Neuronen erkennbar. Das bedeutet, dass diese erhöhte Varianz nicht auf den Datensatz zurückzuführen ist, sondern auf das Verfahren selbst.
Für die Datenunsicherheit kann zusätzlich deren Kalibrierung untersucht werden. Abbildung 5.26 zeigt hierfür die Kalibrierungskurven des besten Modells mit 1024 bzw. 32 Neuronen für jeden Ausgabekanal. Dabei ist leicht erkennbar, dass für das Modell mit 1024 Neuronen grundsätzliche für alle Ausgabekanäle eine gute Kalibrierung vorliegt. Zudem scheint die Güte der Kalibrierung unabhängig von dem Fehler, den das Modell auf einem Ausgabekanal macht. Wie zuvor in Abbildung 5.21 (Seite 94) erkennbar war, macht das Modell beispielsweise auf dem Ausgabekanal 0 einen deutlich größeren Fehler als auf Ausgabekanal 6. Dennoch scheint die Datenunsicherheit auf diesem Kanal

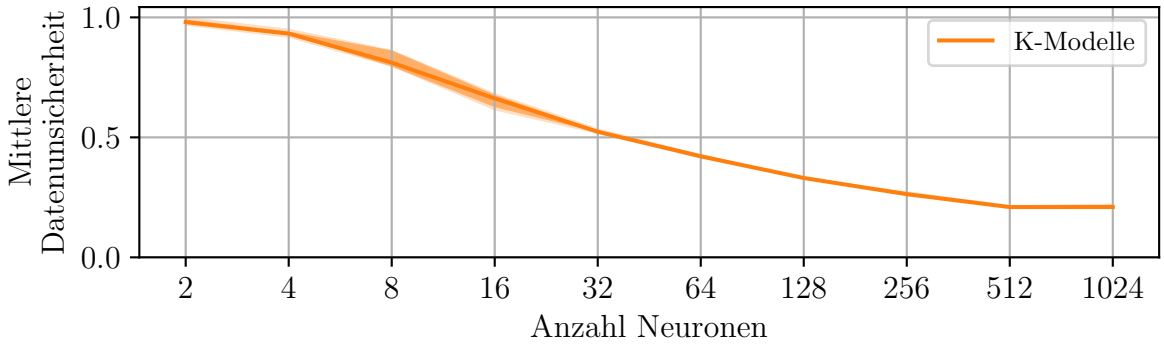



**Abbildung 5.25:** Verlauf der Datenunsicherheit in Abhängigkeit der Modellgröße
*Die Abbildung zeigt den Verlauf der Datenunsicherheiten als Funktion der Modellgröße. Dabei sind die Ergebnisse für N-Modelle und K-Modelle abgetragen.*

besser kalibriert.
Diese Beobachtung deckt sich mit den Ergebnissen auf den synthetischen Daten. Dort wiesen die Datenunsicherheiten der Modelle mit 32 Neuronen auch eine gute Kalibrierung auf, obwohl diese zu klein sind, um die Kosinusfunktion abbilden zu können.
Das Modell mit 32 Neuronen weist in den meisten Fällen eine schlechtere Kalibrierung auf, jedoch ist die Kalibrierung in vielen Fällen vergleichbar zu der Modells mit 1024 Neuronen. Bei einem Vergleich mit den Beispielkurven aus Abschnitt 5 (Seite 35) kann vermutet werden, dass die kleineren Modelle grundsätzlich eine größere Datenunsicherheit schätzen und einen größeren Fehler auf zu den Zielwerten machen. Die Tatsache, dass das Modell mit 32 Neuronen einen MSE von 0.262 und das Modell mit 1024 Neuronen einen Wert von 0.086 besitzen, unterstützt diese Vermutung, da die Kurven für einen alleinigen Unterschied im Fehler anders verlaufen würden.

**Fazit**

In diesem Abschnitt wurden grob die Unsicherheiten allgemein betrachtet. Dabei konnten ähnliche Beobachtungen, wie bei den Experimenten auf synthetischen Daten, gemacht werden. Dabei wurde der Verlauf der beiden Unsicherheiten in Abhängigkeit der Modellgröße untersucht. Zusätzlich wurde die Kalibrierung der Datenunsicherheit für Modelle verschiedener Größe betrachtet.

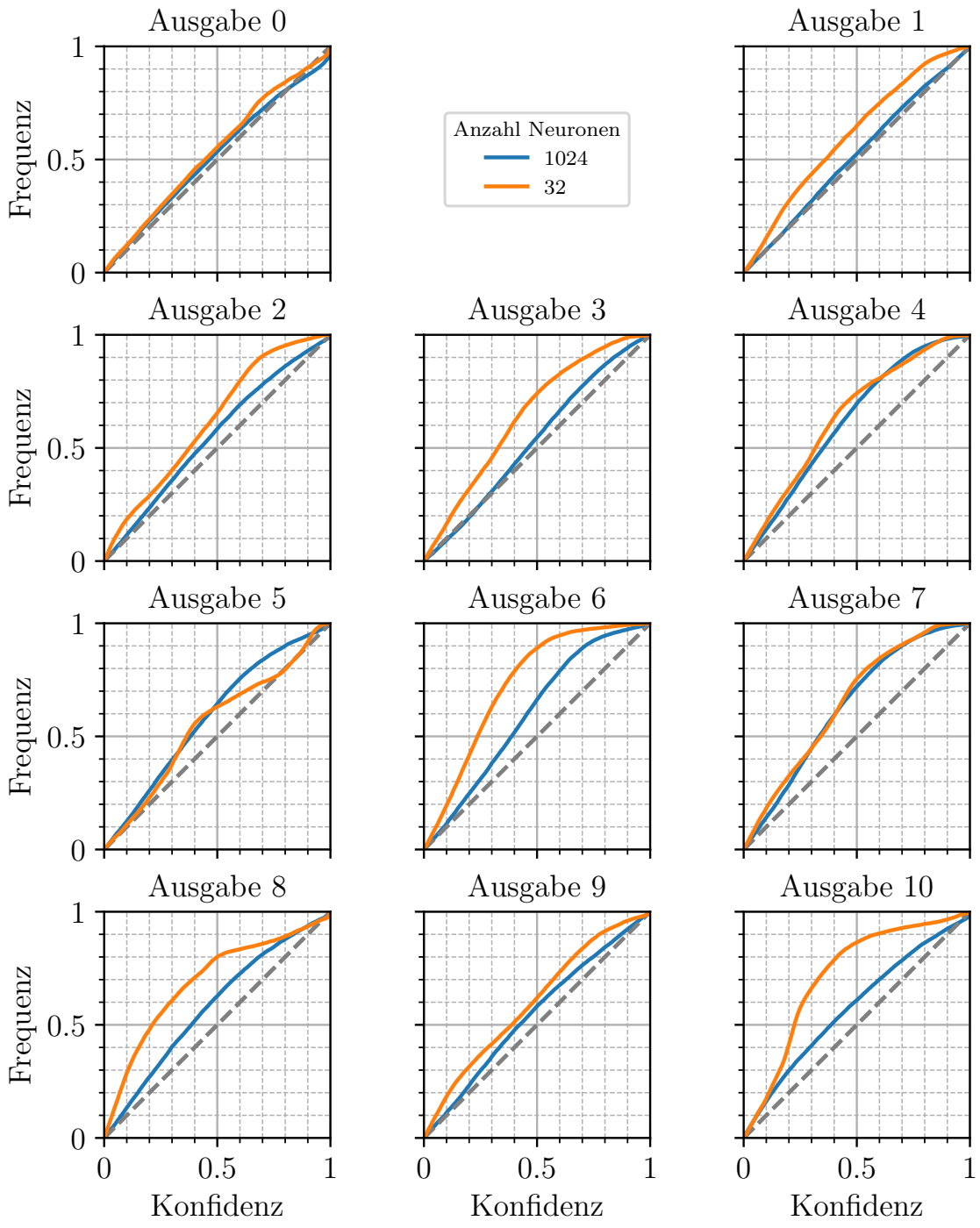



**Abbildung 5.26:** Kalibierungskurven für alle Ausgabekanäle

*Zu sehen sind Kalibrierungskurven für alle Ausgabekanäle. Bei den verwendeten Modellen handelt es sich um jeweils das Beste über den Durchläufen.*

### 5.2.3 Anomaliedetektion

Im Gegensatz zu den Untersuchungen auf den synthetischen Daten, liegt der Fokus bei den Daten der SES auf der Detektion der Anomalien. Anhand der zur Verfügung stehenden Labelinformationen werden in diesem Abschnitt die beiden Arten der Unsicherheiten anhand ihrer Eignung zur Trennung von Anomalien und normalen Daten bewertet. Wie im Abschnitt 4.2 (Seite 50) beschrieben, werden hierfür unterschiedliche Herangehensweisen, basierend auf Annahmen an die Anomalien, untersucht.
In den nächsten Abschnitten werden diese Herangehensweisen, abhängig von der verwendeten Unsicherheit, nochmals kurz beschrieben und deren Ergebnisse präsentiert. Bedingt durch den Ablauf der Untersuchungen werden dabei zuerst du Ergebnisse zu den Datenunsicherheiten und anschließend die der Modellunsicherheiten präsentiert. Für eine ausführliche Beschreibung und einer Diskussion der Vor- und Nachteile der beiden Herangehensweisen wird wiederum auf Abschnitt 4.2 verwiesen.

#### Detektion anhand der Datenunsicherheit

Damit Anomalien anhand der Datenunsicherheit detektiert werden können, muss ein Modell für diese eine unterschiedliche Datenunsicherheit schätzen als für die normalen Daten. Für diese Experimente besteht die intuitive Annahmen, dass die Anomalien einen Widerspruch zu den normalen Daten bilden. Das bedeutet eine Anomalie besitzt einen stark unterschiedlichen Zielwert zu einem normalen Datenpunkt mit ähnlicher Eingabe.
Da für das Training sowohl normale Daten als auch Anomalien im Trainingsdatensatz vorhanden sein müssen, kann der Datensatz einfach aus den vorhandenen Daten zusammengestellt werden, ohne auf die Labelinformationen zu achten. Dadurch ist es, im Gegensatz zu der Detektion auf Basis der Modellunsicherheit, möglich ein einzelnes Modell für alle Ausgabekanäle zu trainieren. Die folgenden Modelle besitzen daher insgesamt 22 Ausgabeneuronen, welche aus elf Neuronen für die Schätzung des jeweiligen Zielwertes der Ausgabekanäle und aus elf Neuronen für die jeweilige Datenunsicherheit bestehen. Da die Detektion auf Basis der Datenunsicherheit eine Schätzung dieser voraussetzt, werden im Folgenden nur K-Modelle betrachtet.
Abbildung 5.27 zeigt die ROC-Kurven für Modelle verschiedener Größe.

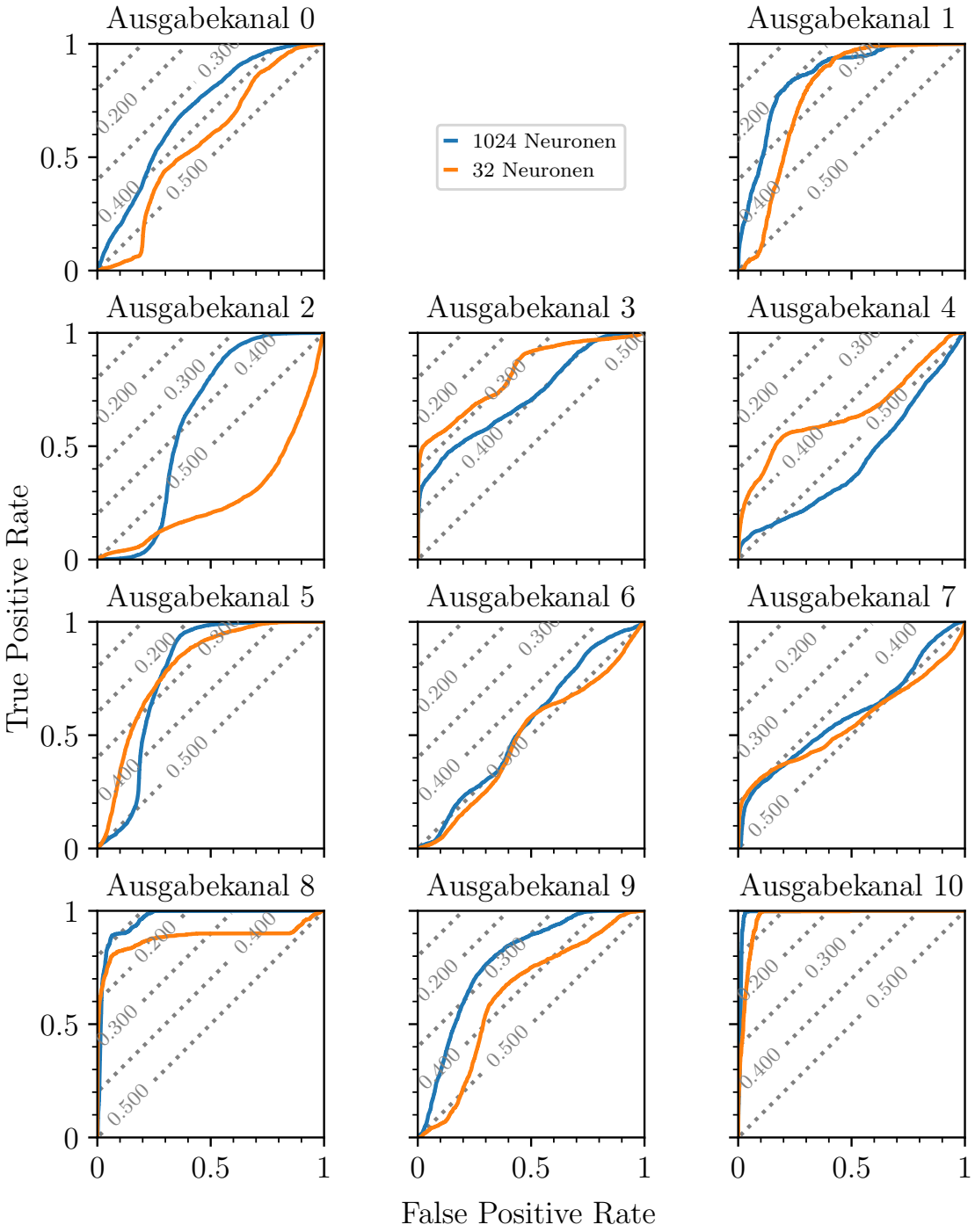



**Abbildung 5.27:** ROC-Kurven für jeden Ausgabekanal auf Basis der Datenunsicherheit

*Abgebildet sind die ROC-Kurven für alle elf Ausgabekanäle auf Basis der Datenunsicherheit. Dabei werden die Ergebnisse für Modelle mit* 32 *und* 1024 *Neuronen gezeigt.*

Modelle mit 32 Neuronen sind in der Praxis nicht relevant, da sich zuvor gezeigt hat, dass diese zu klein sind, um die Ausgabekanäle ausreichend zu beschreiben. Diese werden hier dennoch betrachtet, um zu untersuchen, ob der Fehler eines Modells eine Auswirkung auf die Qualität der Anomaliedetektion hat.
Gut erkennbar sind die mitunter starken Unterschiede zwischen den verschiedenen Ausgabekanälen. Zum Beispiel sind die Ergebnisse für Ausgabekanal 6 auf dem Level einer zufälligen Anomaliedetektion, während die Abbildung für Ausgabekanal 10 nahezu perfekte ROC-Kurven zeigt. Zudem ist erkennbar, dass in den meisten Fällen die ROC-Kurve der Modelle mit 32 Neuronen schlechter ist als die der Modelle mit 1024 Neuronen. Dieses Verhalten ist zu erwarten, da die kleineren Modelle nicht in der Lage waren eine gute Abbildung der Ausgabekanäle zu finden. Daher ist von diesen auch nicht zu erwarten, dass diese eine brauchbare Datenunsicherheit schätzen können. Allerdings gibt es Ausnahmen, wie beispielsweise Ausgabekanäle 3 und 4, für die die kleineren Modelle eine bessere ROC-Kurve ergeben haben. Zudem erreichen die kleineren Modelle für Ausgabekanal 10 nur eine geringfügig schlechtere ROC-Kurve als die Modelle mit 1024 Neuronen.
All dies lässt vermuten, dass die Qualität der Anomaliedetektion in diesen Experimenten weniger von den trainierten Modellen, sondern mehr von den Ausgabekanälen oder den Daten innerhalb des Testsets abhängt. Der nicht vorhandene Zusammenhang zwischen Modellfehler und Qualität der Anomaliedetektion ist auch zu erkennen, wenn die Fehler eines Modells für die verschiedenen Ausgaben mit den dazugehörigen ROC-Kurven verglichen werden. Im vorherigen Abschnitt dieses Kapitels zeigte die Abbildung 5.21 (Seite 94), dass die Modelle mit 1024 Neuronen für den Ausgabekanal 10 einen deutlich höheren Fehler produziert haben, als für die Ausgaben 3 bis 7. Dennoch fallen die ROC-Kurven auf diesen Kanälen deutlich schlechter aus, als für den Ausgabekanal 10.
Speziell die Ergebnisse für Ausgabekanal 10 widersprechen der zuvor getroffenen Annahme an die Anomalien. Falls, wie angenommen, die Anomalien einen Widerspruch zu den normalen Daten bilden, dann würden normale Daten oft als Anomalie klassifiziert, da diese Teil des Widerspruchs sind und somit auch eine erhöhte Datenunsicherheit zugeteilt bekommen würden. Eine ROC-Kurve, wie sie bei Ausgabekanal 10 zu sehen ist, würde allerdings bedeutet, dass sehr wenige Datenpunkte fälschlicherweise als Anomalie klassifiziert werden wurden. Für die Tatsache, dass dem im Falle des

Ausgabekanals 10 nicht so ist, gibt es mehrere Erklärungen. Zum Beispiel könnte bei den Versuchen der Testdatensatz zufällig so gewählt worden sein, dass sich nur normale Daten darin befinden, welche nicht Teil der Widersprüche waren. Somit würden diese auch keine erhöhte Datenunsicherheit zugeteilt bekommen und somit nicht als Anomalie klassifiziert.
Auch könnten sich die Anomalien auf einen bestimmten Bereich im Eingaberaum konzentrieren. Das könnte dazu führen, dass nur ein sehr kleiner Teil der normalen Daten Teil der Widersprüche wäre und somit der überwiegende Anteil der Daten, welche eine erhöhte Datenunsicherheit zugeteilt bekommen, aus Anomalien bestehen würde. Auch könnten die Anomalien unter sich Widersprüche bilden, was eine optimale Gegebenheit für die Anomaliedetektion anhand der Datenunsicherheit darstellen würde, da somit keine oder nur wenige normale Daten fälschlicherweise als Anomalie klassifiziert werden würden. Zudem ist es durchaus plausibel, dass Anomalien unter sich Widersprüche bilden, da es in einem industriellen Betrieb in der Regel viele unterschiedliche Anomalien gibt und sich diese in ihrer Wirkung ausreichend unterscheiden könnten, um die nötigen Widersprüche zu bilden. Zudem muss sich eine Anomalie nicht nur in einem der Ausgabekanäle zeigen, sondern kann mit Änderungen in den Eingabekanälen zusammenhängen. Durch diese Anomalien in den Eingabekanälen können so Eingaben für das Modell entstehen, welche sich von einem normalen Betriebszustand unterscheiden, wodurch diese Datenpunkte keinen Widerspruch zu den normalen Daten bilden würden.

### Detektion anhand der Modellunsicherheit

Für die Anomaliedetektion anhand der Modellunsicherheit wird angenommen, dass sich Anomalien in der Eingabe von normalen Daten unterscheiden. Damit Anomalien von normalen Daten auf Basis der Modellunsicherheit unterschieden werden können, darf ein Modell nur auf Daten einer der beiden Gruppen trainiert werden. Für die folgenden Experimente wurden daher Modelle nur auf normalen Daten trainiert.
Bedingt durch den großen Anteil an Anomalien in den Daten wurden daher einzelne Modelle für jeden der elf Ausgabekanäle trainiert, da der Anteil an Anomalien pro Ausgabekanal wesentlich geringer ist. Das bedeutet in diesem Fall besitzen die N-Modelle ein Ausgabeneuron während die K-Modelle zwei besitzen.
Abbildung 5.28 zeigt die ROC-Kurven der verschiedenen Modelltypen mit 128 Neuronen. Bei einem Vergleich der Kurven für die beiden Modelltypen ist zu erkennen,

dass diese einen sehr ähnlichen Verlauf aufweisen. Der größte Unterschied ist dabei für Ausgabekanal 5 zu sehen. Eine eindeutige Überlegenheit eines der beiden Modelltypen ist zudem nicht erkennbar. Daher kann geschlussfolgert werden, dass die simultane Modellierung der Datenunsicherheit durch das Verfahren nach [Kendall und Gal, 2017] und der Modellierung der Modellunsicherheit nach [Gal und Ghahramani, 2016b] keine starken Auswirkungen auf die Anomaliedetektion durch die Modellunsicherheit hat.

Ein weiterer wichtiger Punkt ist die Ähnlichkeit der ROC-Kurven zu denen aus Abbildung 5.27. Ausgabekanal 1 bildet dabei eine Ausnahme, da hier die ROC-Kurven für die Datenunsicherheit deutlich besser sind als die Kurven der Modellunsicherheit. Eine Ähnlichkeit der ROC-Kurven suggeriert, dass sich die beiden Typen an Unsicherheiten für die Daten ähnlich verhalten. Dies würde bedeutet, dass möglicherweise eine Mischung aus den beiden Grundannahmen der verschiedenen Herangehensweisen das reale Verhältnis zwischen normalen Daten und Anomalien beschreibt.

Für Kanäle bei denen weder durch die Datenunsicherheit noch durch die Modellunsicherheit eine Erkennung der Anomalien erreicht werden konnte, würde dies bedeutet, dass der Schnitt im Eingaberaum zwischen den normalen Daten und den Anomalien zu groß ist. Somit erhalten die Anomalien und die normalen Daten ähnliche Werte für die Modellunsicherheit und die Datenunsicherheit, was zu einer zufälligen Anomaliedetektion führt.

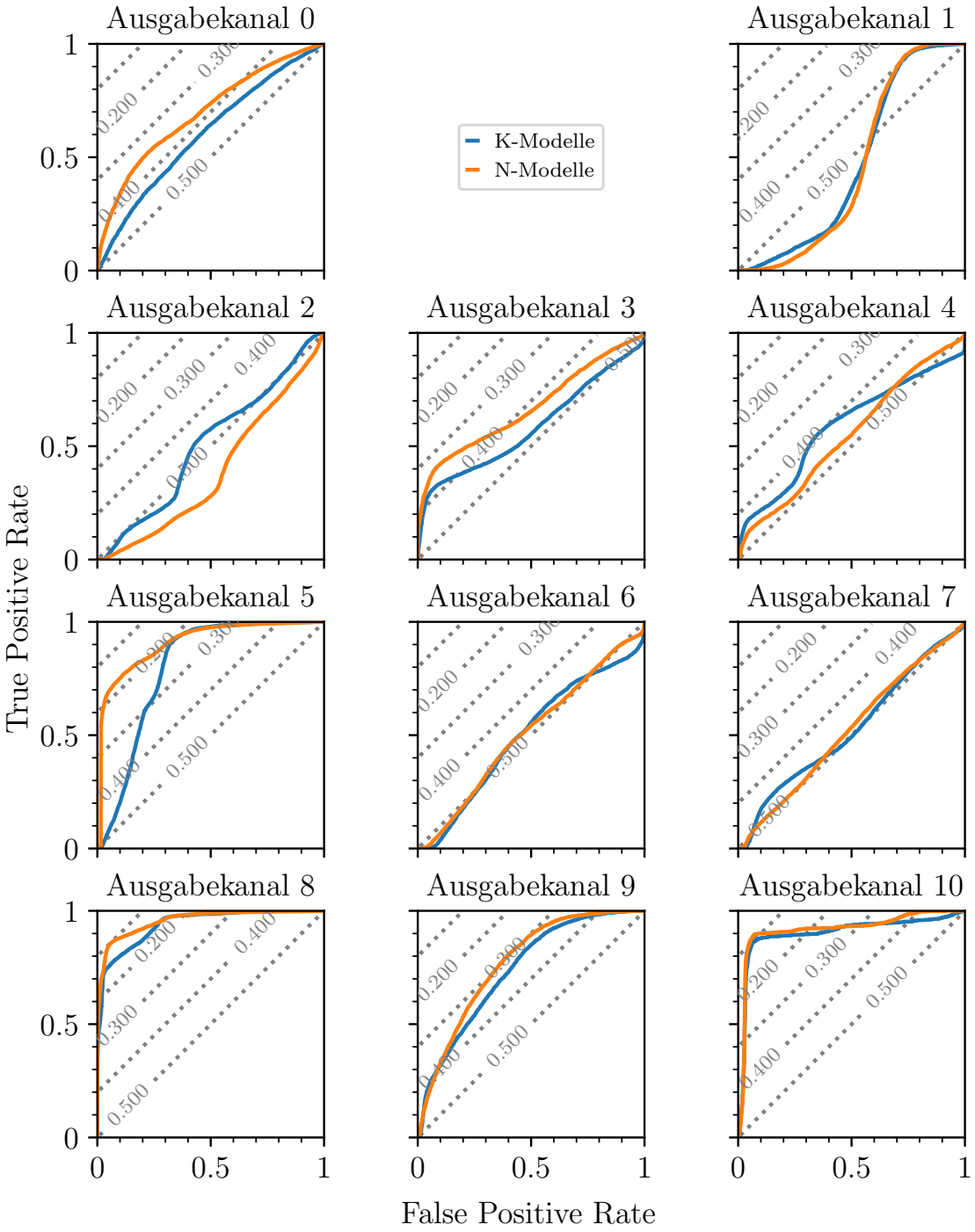



**Abbildung 5.28:** ROC-Kurven für jeden Ausgabekanal auf Basis der Modellunsicherheit

*Abgebildet sind die ROC-Kurven für alle elf Ausgabekanäle auf Basis der Modellunsicherheit. Dabei werden jeweils die Ergebnisse der N-Modelle und die K-Modelle mit* 128 *Neuronen gezeigt.*

**Fazit**

In diesem Abschnitt wurden die beiden Unsicherheitstypen daraufhin untersucht, ob diese dazu geeignet sind Anomalien zu erkennen. Dabei hat sich gezeigt, dass beide Unsicherheiten oft ähnliche Ergebnisse liefern. Allerdings sind diese nicht konsistent über den verschiedenen Ausgabekanälen. Es kann im Allgemeinen keine gute Trennung der Anomalien von den normalen Daten vorgenommen werden. Grund hierfür könnten die Labelinformationen, welche zur Evaluation verwendet wurden, sein, da diese nicht der Wirklichkeit entsprechen müssen, sondern selbst durch ein Verfahren zur Anomaliedetektion generiert wurden. Das bedeutet, es ist möglich, dass die Verfahren dieses Abschnitts Anomalien richtig erkennen, die allerdings von dem Verfahren der SES nicht erkannt wurden. Die schlechte Qualität der Anomaliedetektion kann allerdings auch daran liegen, dass die Anomalien, so verteilt sind, dass diese schlecht trennbar sind. Dieser Sachverhalt wird der Zusammenfassung der Untersuchungen auf den Daten der SES erneut aufgegriffen und näher erläutert.

## 5.3 Zusammenfassung

Um einen besseren Überblick über die wichtigsten Ergebnisse der Experimente zu erhalten, werden im folgenden Abschnitt die wichtigsten Erkenntnisse zusammengefasst. Speziell für die Untersuchungen auf den Daten der SES werden zusätzliche Schlussfolgerungen im Bezug auf die Anomalien getroffen.

### 5.3.1 Zusammenfassung - Synthetische Daten

Das Ziel der Untersuchungen auf den synthetischen Daten lag darin ein Grundverständnis für die verschiedenen Arten der Unsicherheit zu erlangen. Dafür wurden Modellunsicherheiten und Datenunsicherheiten in Abhängigkeit verschiedener Parameter untersucht und bewertet.
Für die Modellunsicherheit hat sich dabei herausgestellt, dass diese relativ unabhängig von der zu lernenden Funktion ist. Das bedeutet, dass die verwendete Kosinusfunktion gegen eine beliebige Funkion ausgetauscht werden kann, ohne dass dies Auswirkungen auf die Ergebnisse zu der Modellunsicherheit hat. Diese Erkenntnis wurde auf Basis der Ergebnisse von Modellen getroffen, welche zu klein sind, um die Kosinusfunktion

abzubilden. Um diese Behauptung zu belegen wären also weitere gezielte Experimente nötig.
Für die Datenunsicherheiten hat sich gezeigt, dass diese gut kalibriert sind. Dies gilt auch für Modelle, welche zu klein sind, um eine gute Schätzung der Kosinusfunktion zu bilden. Daher kann erwartet werden, dass dieses Verfahren unabhängig des verwendeten Modells gut kalibrierte Unsicherheiten liefert.
Neben diesen Beobachtungen kann anhand der durchgeführten Experimente und deren Ergebnissen ein grundlegendes Verständnis über das Verhalten der Unsicherheiten in Bezug auf Parameter, wie der Modellgröße, erlangt werden.

### 5.3.2 Zusammenfassung - STEAG

Die Experimente auf den Daten der SES haben wichtige Erkenntnisse im Bezug auf das Training eines Modells auf den Prozessdaten und für die Anomaliedetektion anhand der Unsicherheiten ergeben.
Es hat sich gezeigt, dass ein einziges Modell auf allen Eingabe- bzw. Ausgabekanälen trainiert werden kann und es dabei vergleichbare Ergebnisse zu den kleineren Modellen der SES liefert. Der entscheidende Vorteil dabei ist, dass keine Vorauswahl der Eingabekanäle nötig ist und damit kein Vorwissen über die Zusammenhänge zwischen den einzelnen Messgrößen vorhanden sein muss.
Bei der Anomaliedetektion anhand der Unsicherheiten hat sich gezeigt, dass dies im Allgemeinen keine guten Ergebnisse liefert. Allerdings können durch die Experimente Rückschlüsse auf die Eigenschaften der Anomalien geschlossen werden. Da keine der beiden Arten von Unsicherheit dazu in der Lage ist, die Anomalien zu erkennen, scheint es wahrscheinlich, dass in vielen Fällen eine Mischform der beiden Annahmen an die Anomalien zutrifft.
Für Fälle in denen die Klassifikation für beide Unsicherheitstypen einem Zufallsprozess gleicht, kann zudem ein Worst Case definiert werden. Abbildung 5.29 zeigt, analog zu den Abbildungen 4.6 und 4.4, die hypothetische Verteilung von Anomalien und normalen Daten. In dem dort beschriebenen Fall könnten die Anomalien weder anhand der Datenunsicherheit noch anhand der Modellunsicherheit erkannt werden. Dies liegt daran, dass die Anomalien den gleichen Bereich im Eingaberaum abdecken wie die normalen Daten.

Diese Problematik entsteht allerdings nur, da eine Unterteilung der Messgrößen in Eingabe und Ausgabe vorgenommen wurde. Die Anomalien des hypothetischen Systems in Abbildung 5.29 könnten leicht von den normalen Daten getrennt werden, wenn neben der eigentlichen Eingabe $x$ zusätzlich $y$ als Eingabe des Modells bereitstehen würde. Für die Daten der SES würde dies bedeuten, dass die elf Ausgabekanäle als zusätzliche Eingabekanäle dienen würden. Da sich in den Untersuchungen gezeigt hat, dass für die Modellunsicherheit die eigentliche Funktion nicht relevant ist, kann dann eine zufällig gewählte Funktion Zielwerte für eine einzelne Modellausgabe liefern. Beispielsweise könnte ein zufällig initialisiertes Neuronales Netz als Beschreibung einer zufälligen Funktion dienen. Dies hat zudem den Vorteil, dass keine Suche nach einer geeigneten Modellarchitektur nötig ist, da einfach die des zufälligen Neuronalen Netzes verwendet werden kann.
Durch dieses vorgehen könnten dann die Anomalien anhand der Modellunsicherheit erkannt werden, wenn die Modelle ausschließlich auf normalen Daten trainiert werden. Nachteil an diesem Vorgehen wäre allerdings, dass keine Anomaliedetektion pro Messgröße mehr möglich ist.

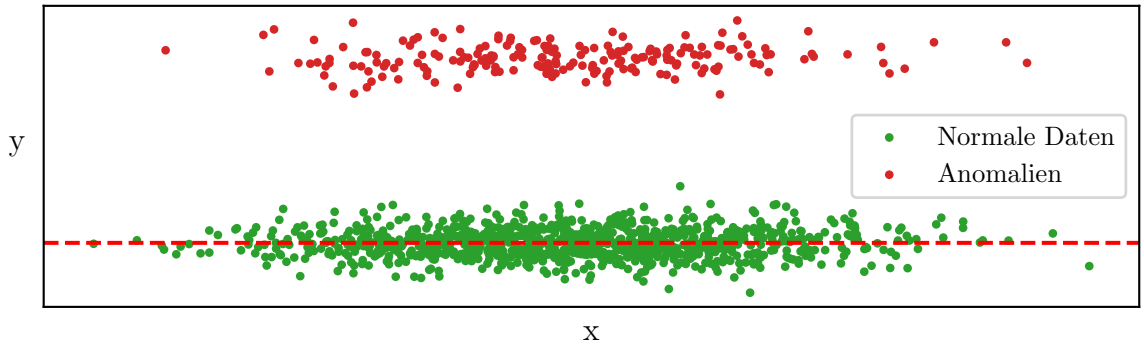



**Abbildung 5.29:** Darstellung eines Worst Case für die Anomaliedetektion anhand der Unsicherheiten
*Dargestellt ist der Worst Case für die Detektion der Anomalien anhand der Unsicherheiten. In diesem Fall ist der Schnitt der Bereiche des Eingaberaums für Anomalien und normale Daten maximal.*

# Kapitel 6

# Zusammenfassung und Ausblick

Nachdem nun die durchgeführten Experimente und deren Ergebnisse präsentiert wurden, wird in diesem Kapitel abschließend ein Überblick über diese Masterarbeit gegeben. Zudem werden Ansatzpunkte für weiterführende Arbeiten und zusätzliche Experimente aufgezählt.

## 6.1 Zusammenfassung

Ziel dieser Masterarbeit war es eine Grundlage für eine Arbeit mit Unsicherheiten zu bilden. Dazu gehört die Aufarbeitung des State of the Art für die Berechnung von Unsicherheiten und eine grundlegende Untersuchung der verschiedenen Unsicherheiten. Hierfür wurde ein synthetischer Datensatz generiert, anhand dessen die verschiedenen Unsicherheiten bewertet werden können.
Für die Untersuchungen der Modellunsicherheit wurde dabei das Verfahren nach [Gal und Ghahramani, 2016b] mit dem Sampling-Verfahren nach [Hoffman und Gelman, 2014] verglichen. Zudem wurde untersucht inwieweit unbekannte Daten anhand der berechneten Modellunsicherheit erkannt werden können. Dabei hat sich gezeigt, dass alle Modellunsicherheiten eine gute Klassifikation der unbekannten Daten ermöglicht haben.
Das Verfahren nach [Kendall und Gal, 2017] zur Schätzung von Datenunsicherheit wurde anhand der Kalibrierung bewertet. Dabei hat sich gezeigt, dass die geschätzten Datenunsicherheiten für verschiedene Modelle gut kalibriert sind. Damit kann erwartet werden, dass dieses Verfahren unabhängig der Modellarchitektur funktioniert.

Zusätzlich wurde die Verwendung von Unsicherheiten in der Problemstellung der Anomaliedetektion untersucht. Dabei wurde auf Basis eines Datensatzes der STEAG Energy Services GmbH (SES) untersucht, inwiefern es möglich ist anhand von Unsicherheiten Anomalien von normalen Daten zu unterscheiden. Eine ausreichende Erkennung der Anomalien ist hier im allgemeinen nicht gelungen, allerdings konnten anhand der Ergebnisse Schlussfolgerungen über die Eigenschaften der Anomalien getroffen werden. Ob die untersuchten Verfahren für die praktische Anwendung geeignet sind müsste allerdings genau getestet werden.

## 6.2 Ausblick

Da für diese Arbeit eine Auswahl der zu untersuchenden Verfahren getroffen werden musste, könnten übrige Verfahren analog untersucht werden. Besonders die Verfahren aus dem Bereich der Ensembles ist dabei von Interesse, da der Fokus dieser Arbeit auf den Verfahren lag, welche die bayessche Theorie als Grundlage haben. Zudem sind die Gauß-Prozesse von Interesse, da diese ein beliebtes Modell darstellen, welches auch für tiefe hierarchische Modelle verwendet wird.
Weiterhing ist eine genauere Untersuchung des Verfahrens nach [Kendall und Gal, 2017] in Bezug auf die Berechnung der Datenunsicherheit möglich. Zwar wurde im Laufe dieser Arbeit untersucht welchen Unterschied es macht, wie eine Modellausgabe unter Verwendung von Dropout berechnet wird, allerdings kann diese Untersuchung analog für die Ausgabe der Datenunsicherheit vorgenommen werden. Dies ist wichtig für den Fall, dass nur die Datenunsicherheit von Interesse ist und keine Stichproben der Subnetze berechnet werden können, da dies möglicherweise einen zu großen Rechenaufwand bedeuten würde, oder da kein Dropout beim Training verwendet wurde.
Im Bezug zu dem Verfahren nach [Gal und Ghahramani, 2016b] kann zudem die Erweiterung nach [Gal et al., 2017] untersucht werden, da dieses theoretische Probleme löst und damit die Antwort auf Kritik an MC-Dropout ist.
Ein vielversprechender Ansatz, welcher für die Problemstellung der Anomaliedetektion verwendet werden kann, ist bereits im Abschnitt 5.3.2 beschrieben worden. Anhand der Modellunsicherheit eines Modells, welches alle Messgrößen als Eingabe erhält und anhand dieser eine zufällig gewählte Funktion abbilden soll, könnte eine bessere Trennung der Anomalien von den normalen Daten erreicht werden. Dieser Ansatz ist

dabei einfach umzusetzen und bietet den Vorteil, dass die einzige Annahme an die Anomalien ist, dass sich diese abseits der normalen Daten befinden. Diese Annahme sollte dabei für alle Arten von Anomalien gelten, da diese sich ansonsten nicht von normalen Daten unterscheiden würden und somit untrennbar wären. Nachteil dieses Ansatzes ist allerdings, dass keine Anomaliedetektion pro Eingabekanal vorgenommen werden kann.

Zusätzlich könnte untersucht werden inwieweit sich die verwendeten Verfahren zur Anomaliedetektion im realen Betrieb verhalten, da zumindest für manche Messgrößen eine Anomaliedetektion möglich war. Hierfür müsste geklärt werden, ob im Vorfeld entschieden werden kann, ob eine Erkennung der Anomalien für einen Messwert möglich ist. Zudem könnte ein Unterschied zwischen einem statischen Datensatz, welcher sich über mehrere Jahre erstreckt, und dem realen Betrieb bestehen. Zum Beispiel könnten Temperaturen, welche im Sommer normal sind im Winter als anormal gelten. Da ein Modell hierfür Informationen über die Jahreszeit bräuchte könnten solche Anomalien nicht erkannt werden, während ein Modell, welches auf aktuellen Messdaten nachtrainiert wird, dazu in der Lage sein könnte.

Da der Fokus dieser Arbeit auf der Regression lag, können zudem Untersuchungen für Problemstellungen der Klassifikation vorgenommen werden. Besonderheit dabei stellt die Verwendung der Softmax-Ausgabe dar, da diese in gewissem Maße Datenunsicherheit abbildet. Dies liegt daran, dass bei widersprüchlichen Labelinformationen der Mittelwert einer Gleichverteilung über bestimmten Labeln entspricht. Das bedeutet, dass theoretisch Widersprüche zu einer geringeren Softmax-Ausgabe führen.

# Anhang A

# Ergänzende Unterlagen

## A.1 Synthetische Daten

### A.1.1 Lernkurven der Modelle

In diesem Teil werden repräsentative Beispiele der Lernkurven der trainierten Modelle präsentiert. Dies hat den Sinn, dass die Wahl der maximalen Epochen und des Parameters für das EarlyStopping nachvollzogen werden können.

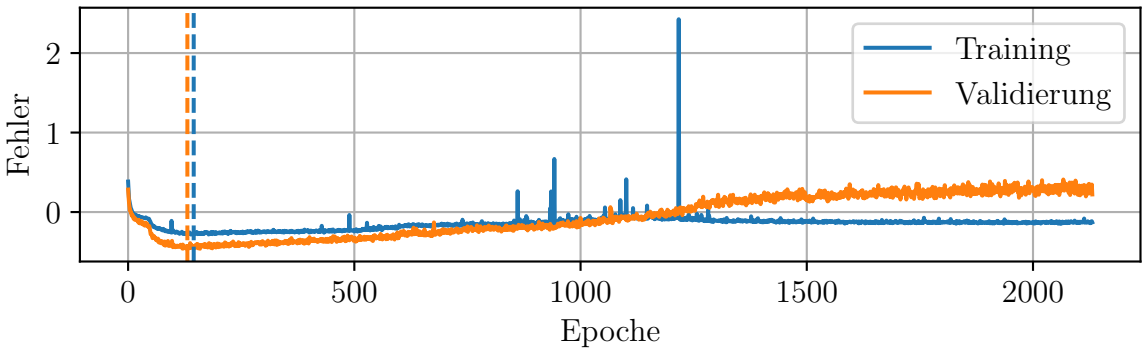



**Abbildung A.1:** Beispiel einer Lernkurve
*Die Abbildung zeigt den Verlauf des Fehlers auf den Trainings- bzw. Validierungsdaten des K-Modells mit* 256 *Neuronen, einer Dropout-Rate von* $p = 0.5$ *und auf varianzheterogenen Daten. Die vertikalen Linien markieren das jeweilige Minimum.*

### A.1.2 Auswirkungen der Aktivierungsfunktion auf Modellunsicherheiten

Da in für die Experimente dieser Masterarbeit nur ReLU-Aktivierungen verwendet worden sind, die Art der Aktivierungsfunktion allerdings eine Auswirkung auf die resultierenden Modellunsicherheiten hat, wird in diesem Abschnitt ein grober Vergleich zwischen verschiedenen Aktivierungen gezogen. Dabei wurde neben ReLU die Sigmoid- und TanH-Funktion untersucht. Abbildung A.2 zeigt den Verlauf der Modellunsicherheit für die verschiedenen Aktivierungen. Für jedes Modell ist zusätzlich der Mean Squared Error (MSE) in der Legende angegeben, da ein großer Unterschied zwischen ReLU-Aktivierung und den andern beiden Funktionen besteht. Für eine Betrachtung der Modellunsicherheit ist dies allerdings nebensächlich, da hier nicht der Wert der Unsicherheit betrachtet werden soll, sondern der Verlauf.

Es ist leicht ein Unterschied zwischen ReLU und den anderen beiden Funktionen erkennbar. Sowohl die Sigmoid- als auch die TanH-Funktion führt zu Modellunsicherheiten, welche für unbekannte Datenbereiche nicht dauerhaft ansteigt. Für diese beiden Aktivierungen saturieren die Modellunsicherheiten ab einem gewissen Wert. Dies ist leicht anhand der Form dieser beiden Funktionen zurückzuführen. Da diese Aktivierungen, im Gegensatz zu der ReLU-Aktivierung, ab einer gewissen Eingabe konstant bleiben, überträgt sich dieses Verhalten auf die gelernte Funktion und damit auf die Modellunsicherheit, da diese anhand der verschiedenen Ausgaben der Subnetze berechnet wird.

Bei Verwendung der Aktivierungsfunktion ist also darauf zu achten, welche Auswirkungen diese auf die resultierenden Modellunsicherheiten hat. Dabei sei angemerkt, dass bei verschiedenen Experimenten Modelle mit einer ReLU-Aktivierung stets besser abgeschnitten haben als mit Sigmoid- oder TanH-Aktivierungen. Das bedeutet, dass in der Regel die Wahl auf die ReLU-Aktivierung fallen sollte, da diese neben einer bessern Modellqualität für eine stetig ansteigende Modellunsicherheit sorgt. Das stellt sicher, dass ab einem gewissen Punkt unbekannte Daten definitiv eine größere Modellunsicherheit verursachen als bekannte Daten.

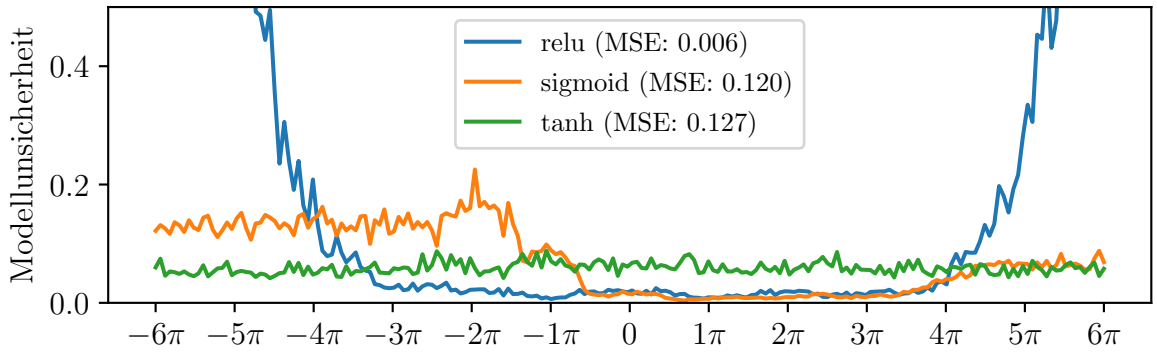



**Abbildung A.2:** Vergleich der Aktivierungsfunktionen

*Zu sehen ist der Verlauf der Modellunsicherheit eines Modells mit verschiedenen Aktivierungsfunktionen.*

### A.1.3 Modellunsicherheit in Abhängigkeit der Sample-Anzahl

In diesem Abschnitt soll nochmals verdeutlicht werden, warum Modellunsicherheiten nicht kalibriert sein müssen. Grund hierfür ist deren Eigenschaft mit der Anzahl an Datenpunkten zu sinken. Abbildung A.3 zeigt zur Veranschaulichung die Ausgabe eines linearen Modells, welches anhand der schwarzen Punkte trainiert wurde. Für jede Abbildung ist die genaue Anzahl an Datenpunkten angegeben und es ist zu erkennen, wie die Modellunsicherheit (blaue Fläche) mit steigender Datenanzahl sinkt. Allerdings

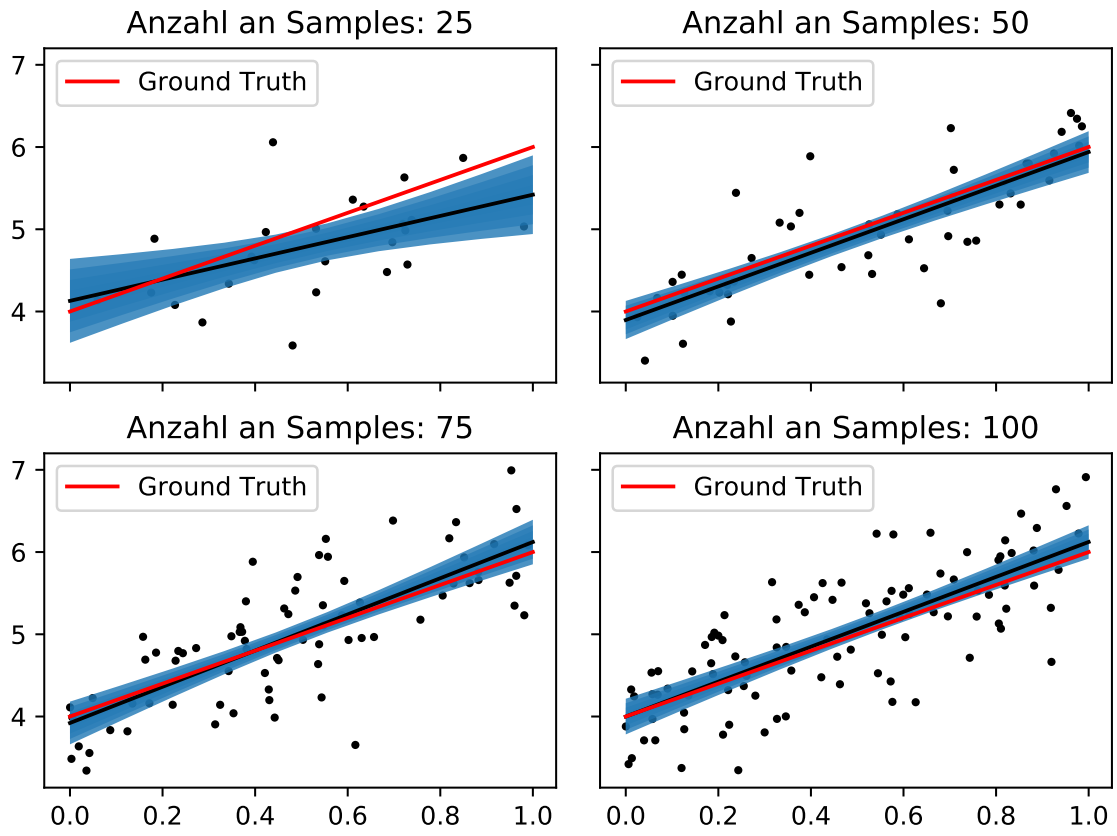



**Abbildung A.3:** Entwicklung der Model-Unsicherheit in Abhängigkeit der Sample-Anzahl.

*Die Grafik zeigt die Ergebnisse des NUTS-Samplers für ein lineares Model in Abhängigkeit der Sample-Anzahl. Punkte sind dabei die verwendeten Trainingsdaten, die rote Linie stellt die unverrauschte Ground Truth dar, die schwarze Linie beschreibt den Mittelwert der Model-Samples und der blaue Bereich stellt die Model-Unsicherheit dar.*

ist auch zu sehen, dass die Datenpunkte verrauscht sind. Diese wurden jeweils entlang der roten Linie gezogen und danach durch mithilfe einer Normalverteilung verrauscht. Es ist erkennbar, dass dieser Rauschterm für jede Abbildung unverändert bleibt. Das bedeutet, auch die Unsicherheit die aus dem Rauschen resultiert ist konstant und ändert sich nicht für verschieden große Trainingssets. Da die Modellunsicherheit aber abnimmt kann diese die Fehler, welche aufgrund des Rauschens gemacht werden, nicht erklären.

### A.1.4 Concrete-Dropout

In diesem Abschnitt wird das Verfahren Concrete Dropout [Gal et al., 2017] kritisch diskutiert. Die Durchführung einfacher Versuche unter Verwendung von Concrete Dropout hat ergeben, dass in vielen Fällen eine sehr geringe Dropout-Rate gelernt wird. Diese befand sich in vielen Fällen im Bereich $< 0.0001$. Diese sehr geringe Dropout-Rate hat Auswirkungen auf die resultierenden Modellunsicherheiten, da allerdings die kleinste in den Experimenten verwendete Dropout-Rate $p = 0.05$ war, ist unklar wie diese Auswirkungen genau aussehen. Zu erwarten ist, dass Modellunsicherheiten generell kleiner ausfallen, was möglicherweise zulasten der Erkennbarkeit von unbekannten Daten geht.
Diese Dropout-Raten können einerseits aus der verwendeten Datenmenge resultieren, da in [Gal et al., 2017] nachgewiesen wurde, dass die gelernte Dropout-Rate mit der Anzahl der Datenpunkte abnimmt und die Datensätze, die in dieser Arbeit verwendet wurden, recht groß sind. Ein weiterer Einfluss könnte allerdings die Verwendung der Reverse KL bei der Optimierung der Dropout-Rate haben.
Abbildung A.4 zeigt einen hypothetischen Fall einer *A-posteriori-Verteilung* des Gewichtes $w$ (blau) und der Verteilung, welche durch Dropout definiert wird (rot). Die Verteilung durch Dropout ist dabei als gemischte Gauß-Verteilung dargestellt, um diese besser darstellen zu können. Für den Fall einer unendlich kleinen Varianz würde die Abbildung der eigentlichen Verteilung entsprechen.
Eine Änderung der Dropout-Rate $p$ würde nun zu einer Umverteilung der Wahrscheinlichkeitsmasse zwischen den beiden Modi $D_1$ und $D_2$ führen. Für $p = 0$ würde dies bedeuten, dass die Wahrscheinlichkeitsmasse bei $D_1$ komplett verschwinden würde. Analog würde für $p = 1$ nur $D_1$ übrig bleiben.

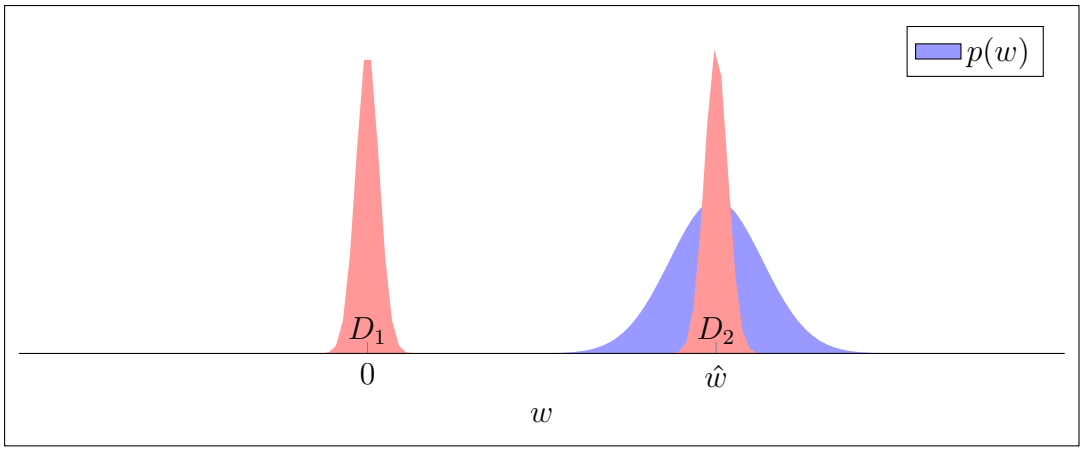



**Abbildung A.4:** Beispiel Optimierung der Dropout-Rate
*Zu sehen ist eine hypothetische A-posteriori-Verteilung des Gewichtes 2 und eine Bernoulli-Verteilung, welche durch die Verwendung von Dropout resultiert.*

Bei einer Optimierung anhand der Reverse KL gilt, wie in Abschnitt 2.4.1 beschrieben, das zero forcing. In diesem Falle würde $D_1$ möglichst klein gehalten werden, was zu einer geringen Dropout-rate führt.
Das bedeutet, falls die Wahrscheinlichkeit mit der das Gewicht $w$ den Wert 0 annimmt sehr gering bzw. gleich Null ist, würde die Optimierung der Dropout-Rate einen sehr geringen Wert ergeben, unabhängig davon wie groß die Varianz der *A-posteriori-Verteilung* $p$ ist.
Da nur eine Dropout-Rate pro Schicht gelernt wird, entspricht der Nullpunkt dem Fall, dass alle Gewichte gleich Null sind. In der Praxis ist zu erwarten, dass eine Optimierung eines Neuronalen Netzes nie alle Gewichte auf Null setzten würde[1], da damit die Ausgabe dieser Schicht konstant wäre. Das bedeutet, es ist davon auszugehen, dass in der Praxis gilt:

$$p(\underline{\mathbf{W}} = \underline{\mathbf{0}} \mid \boldsymbol{D}) = 0 \tag{A.1}$$

wobei $\underline{\mathbf{0}}$ einer Nullmatrix entspricht.
Dies deckt sich mit der Erfahrung, dass die Dropout-Raten, welche anhand von Concrete Dropout optimiert wurden, sehr klein ausfallen.

[1] Ausnahme hierbei bilden besondere Netzwerkarchitekturen wie beispielsweise ResNet [He et al., 2016].

### A.1.5 Auswirkungen der Minibatch-Größe

In Abbildung A.5 ist der Verlauf des MSE in Abhängigkeit der Minibatch-Größe dargestellt. Dabei wurden die Minibatch-Größen $1, 50, 100, 150$ und 200 getestet. Leicht zu erkennen ist, dass große Minibatch-Größen ein besseres Ergebnis liefern als kleine. Damit ist die in den Experimenten verwendete Minibatch-Größe von 200 ausreichen groß für beide Modelltypen.

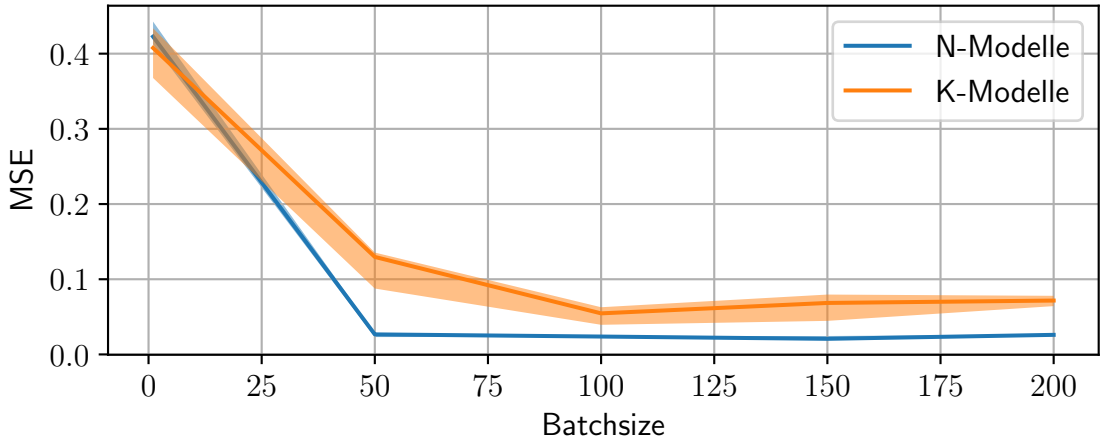



**Abbildung A.5:** Verlauf des MSE in Abhängigkeit der Minibatch-Größe
*Zu sehen ist der Verlauf des MSE für N-Modelle und K-Modelle als Funktion der Minibatch-Größe. Die hier verwendeten Modelle haben* 256 *Neuronen, eine Dropout-Rate von* 0.5 *und wurden auf den varianzhomogenen Daten trainiert.*

### A.1.6 Datensatztests für kleinere Dropout-Raten

In diesem Abschnitt sind die Ergebnisse der Untersuchungen der Datensatzgröße für kleiner Dropout-Raten präsentiert. Abbildung A.6 zeigt die Ergebnisse der Experimente. Dabei ist zusätzlich das Ergebnis mit einer Dropout-Rate von $p = 0.5$ wiederholt, um diese mit den anderen vergleichen zu können. Es ist zu erkennen, dass die Kurven für kleinere Dropout-Raten kaum abhängig von der Datensatzgröße sind. Die Varianz zwischen den Durchläufen und der Fehler nehmen zudem mit der Dropout-Rate ab. Somit ist gezeigt, dass es ausreichend ist sich auf die Ergebnisse der Modelle mit einer Dropout-Rate von $p = 0.5$ zu fokussieren.

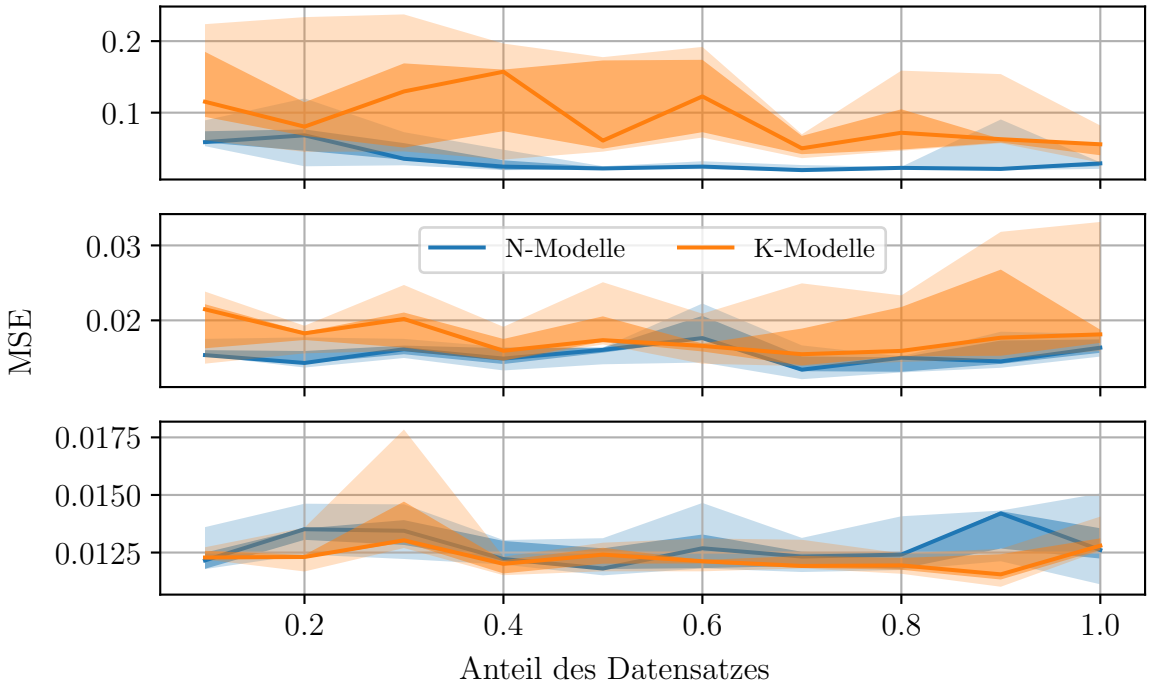


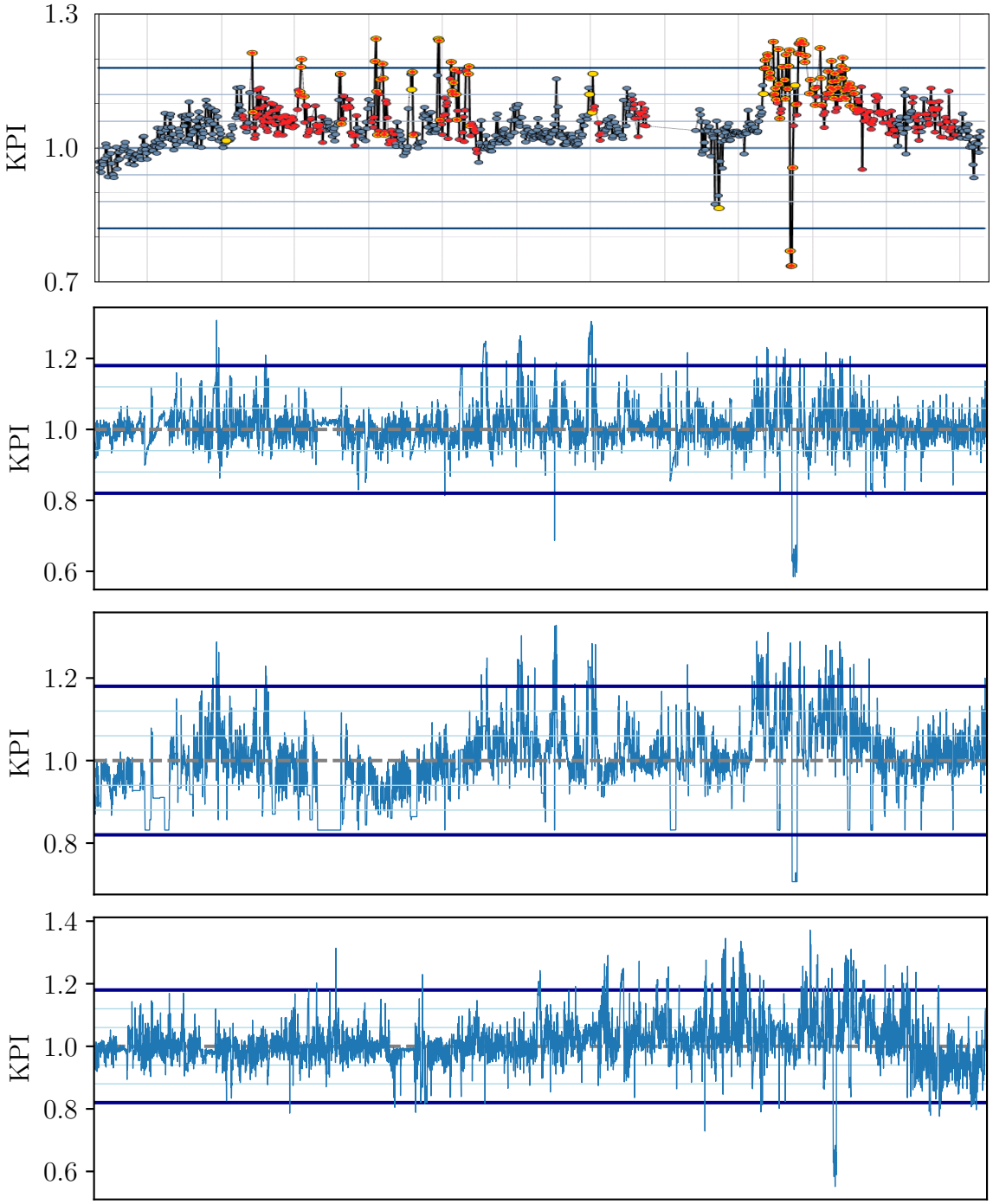

**Abbildung A.6:** Datensatztests für verschiedene Dropout-Raten

*Zu sehen ist jeweils der Verlauf des Fehlers in Abhängigkeit der Größe des Trainingsdatensatzes pro getesteter Dropout-Rate. Es sind Kurven für beide Modelltypen und jeweils die Quantile über fünf Wiederholungen abgetragen.*

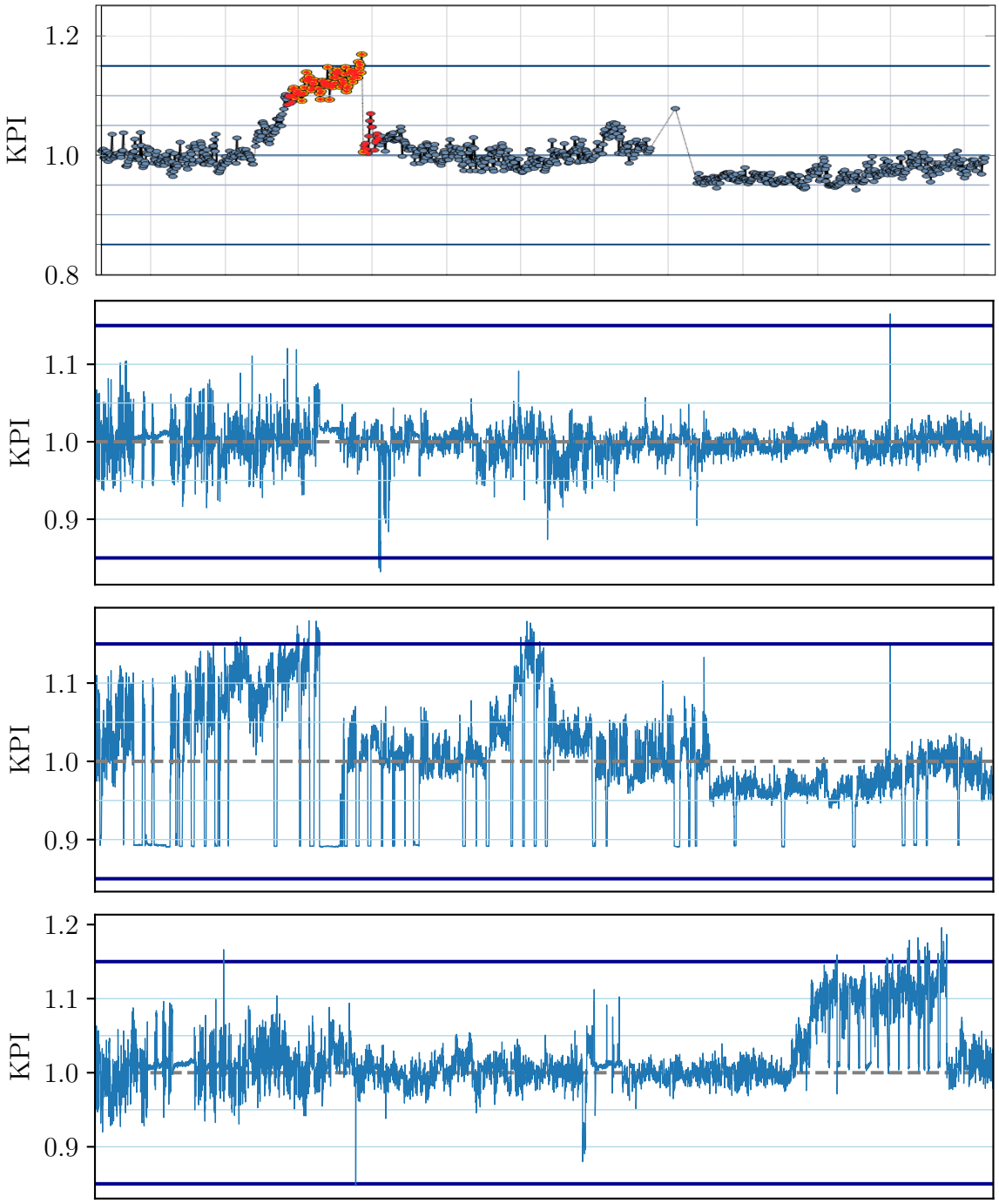
## A.2 STEAG

### A.2.1 STEAG - Modellvergleich

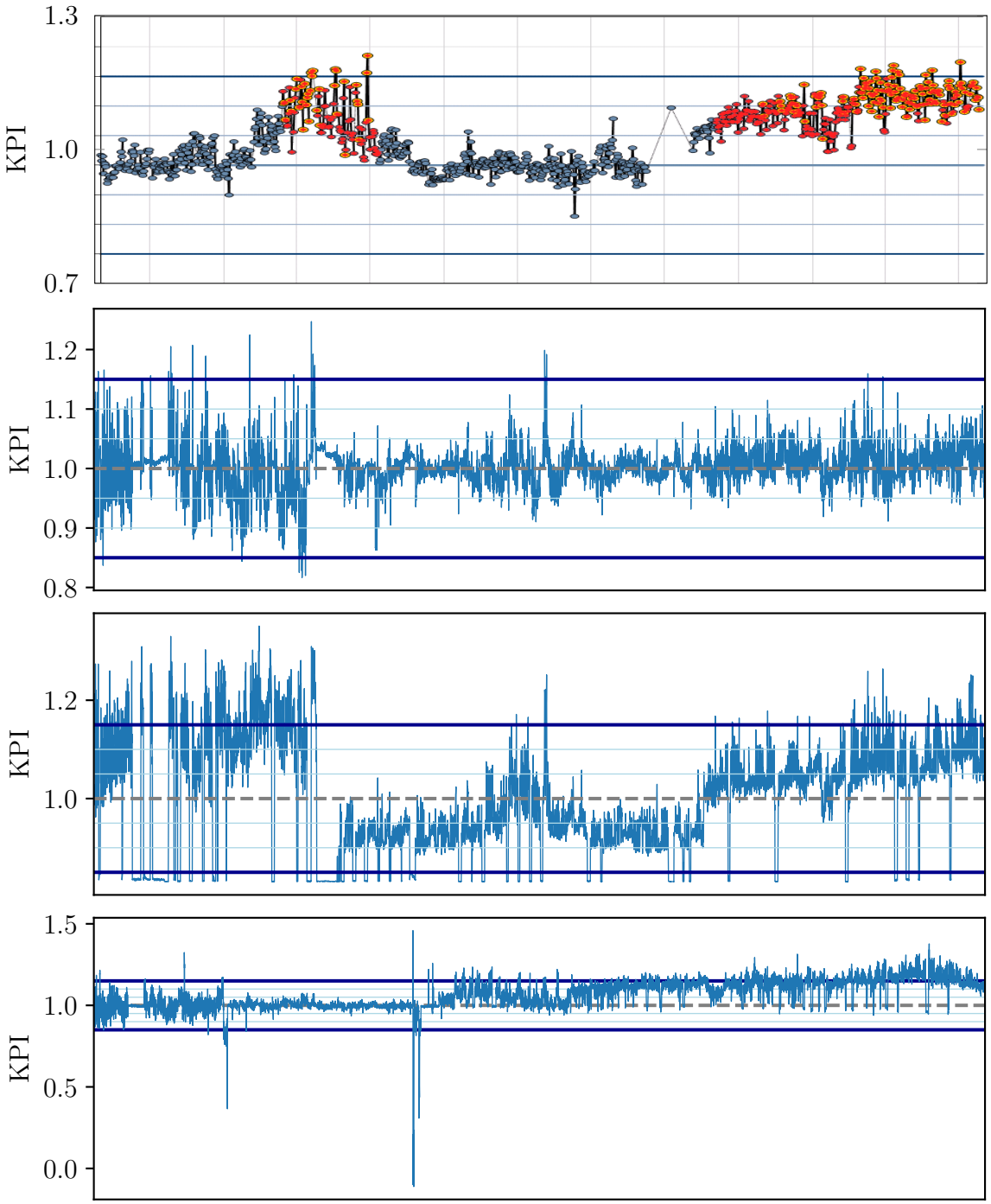
In diesem Abschnitt sind alle Abbildungen der KPI-Werte zum Vergleich der Modelle aufgeführt. Dabei sind neben den Kurven für die Ausgabekanäle 0 und 1 aus Abschnitt 5.2.1 (Seite 95) zusätzlich noch Abbildungen für die Ausgabekanäle 2,5,7 und 10 abgebildet. Zusätzlich sind die Kurven für die Mittelwert-Baseline zu sehen. Für jede Abbildung stellt die obere Kurve das Ergebnis der Modelle der STEAG Energy Services GmbH (SES), die zweite für das Modell mit 1024 Neuronen, die dritte für die jeweiligen Mittelwert-Baseline dar und die letzte die für das Modell, welches nur einen Ausgabekanal lernen musste. Anzumerken ist, dass sich die Datensätze zwischen der ersten, der zweiten bzw dritten und der letzten Grafik unterscheiden.



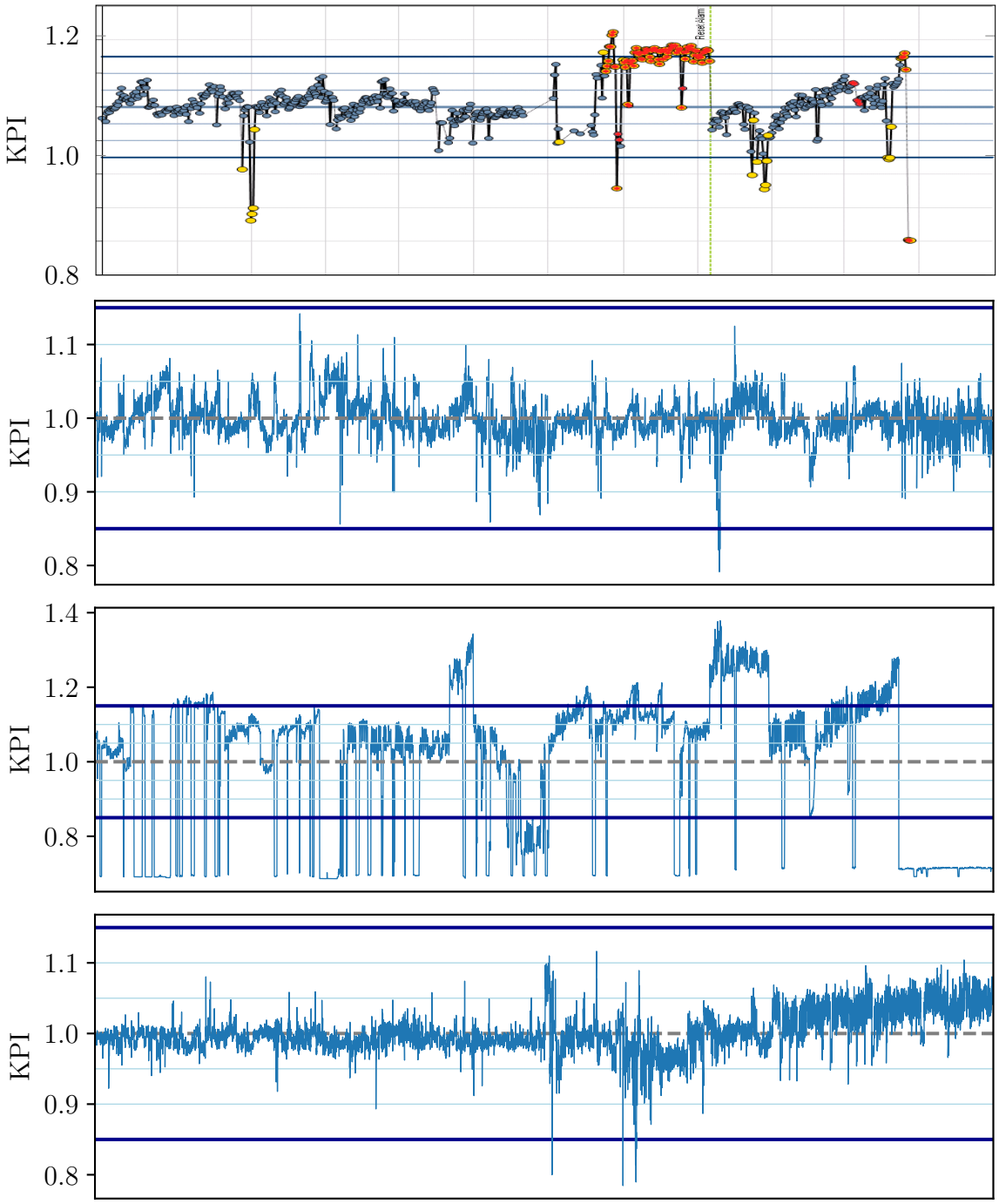

**(a)** Vergleich für Ausgabekanal 0



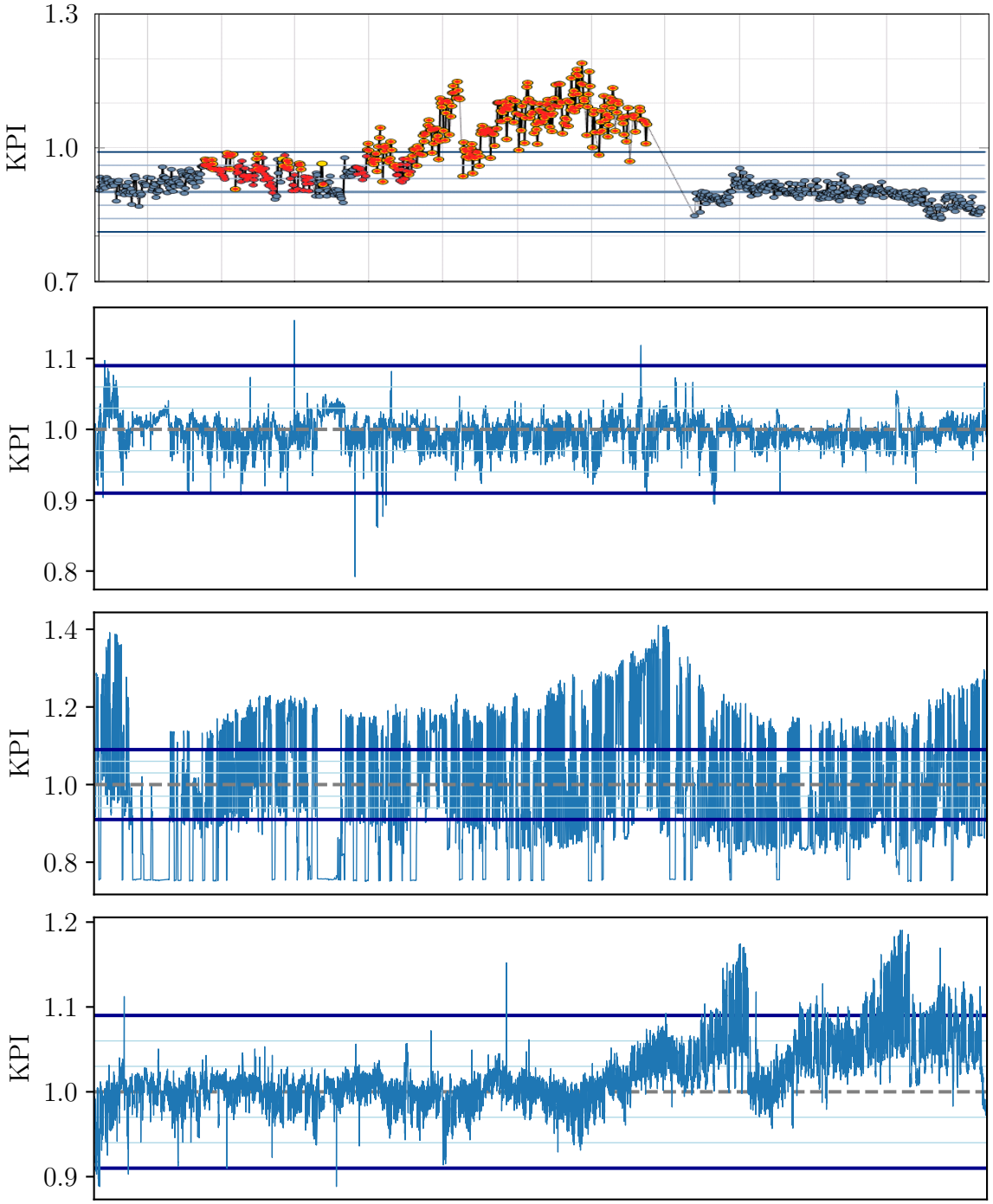

**(a)** Vergleich für Ausgabekanal 1



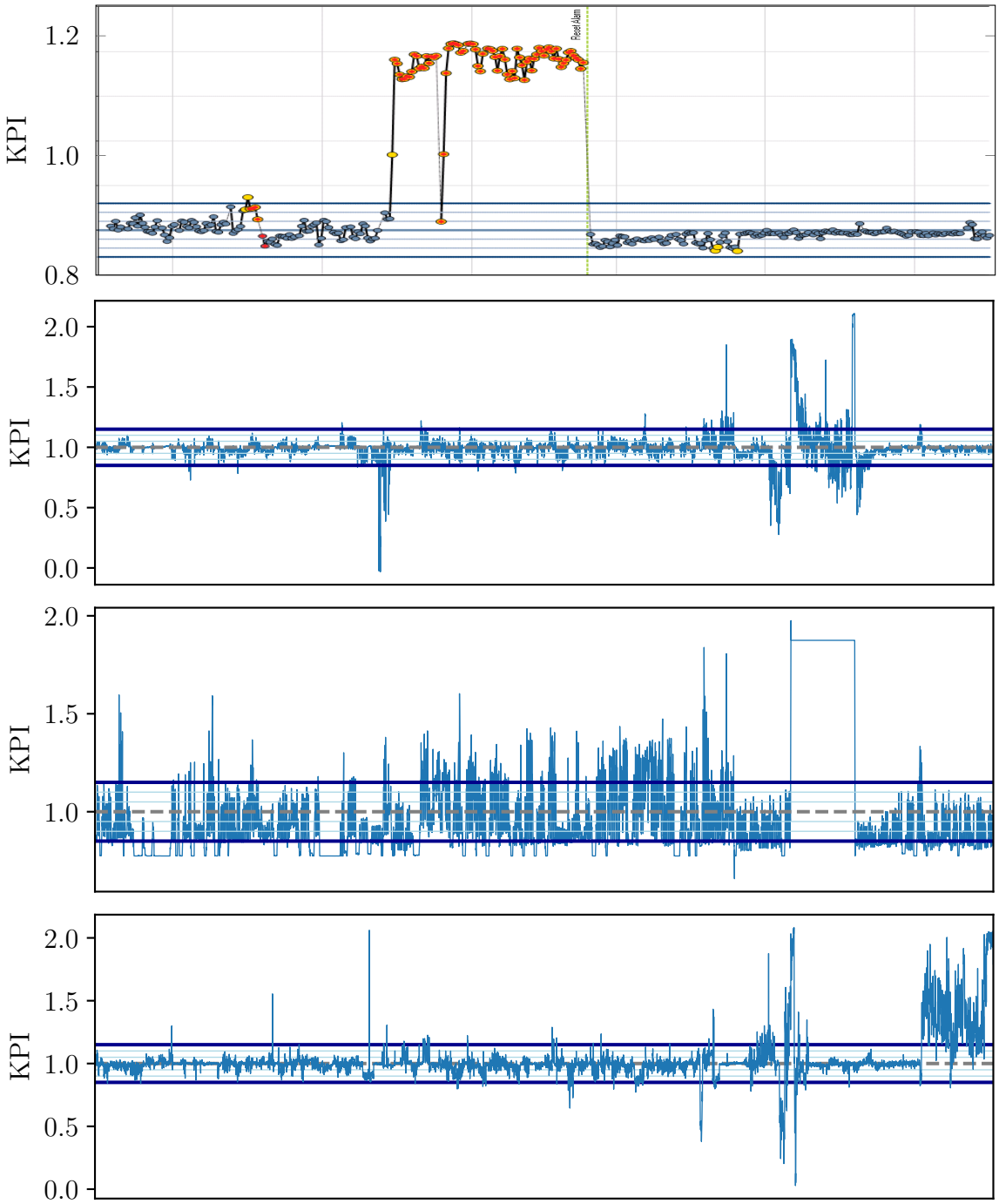

**(a)** Vergleich für Ausgabekanal 2




**(a)** Vergleich für Ausgabekanal 5




**(a)** Vergleich für Ausgabekanal 7




**(a)** Vergleich für Ausgabekanal 10

## A.3 Herleitungen

**Zusammenhang Kreuzentropie / Kullback-Leibler-Divergenz** Die hier angegebene Gleichung zeigt die Herleitung des in Kapitel 2.4.1 beschriebenen Zusammenhangs zwischen der Kreuzentropie $H(\cdot,\cdot)$ und der Forward Kullback-Leibler-Divergenz $D_{KL}(p \,||\, q)$.

$$\begin{aligned}
H(p,q) &= \mathbb{E}_p\Big[-\log q\Big] \\
&= \int_x -p \log q \,\mathrm{d}x \\
&= \int_x p \log \frac{1}{q} \,\mathrm{d}x \\
&= \int_x p(\log \frac{1}{q} + \log \frac{p}{p}) \,\mathrm{d}x \\
&= \int_x p(\log \frac{p}{q} \cdot \frac{1}{p}) \,\mathrm{d}x \\
&= \int_x p \log \frac{p}{q} + p \log \frac{1}{p} \,\mathrm{d}x \\
&= \int_x p \log \frac{p}{q} \,\mathrm{d}x + \int_x p \log \frac{1}{p} dx \\
&= \int_x p \log \frac{p}{q} \,\mathrm{d}x + \int_x -p \log p \,\mathrm{d}x \\
&= D_{KL}(p \,||\, q) + H(p)
\end{aligned} \tag{A.2}$$

# Abbildungsverzeichnis

2.1 Visualisierung der zwei Haupttypen von Unsicherheiten. . . . . 6
2.2 Visualisierung von varianzhomogener bzw. -heterogener Datenunsicherheit. 7
2.3 Darstellung des Prinzips hinter Dropout. . . . . 8
2.4 Aufbau eines ROC-Plots . . . . . 9
2.5 Beispiel einer zu approximierenden Verteilung $p(x)$. . . . . 11
2.6 Beispielhaftes Ergebnis für optimales $q^*(x)$ unter Verwendung der Forward KL. 13
2.7 Beispielhaftes Ergebnis für optimales $q^*(x)$ unter Verwendung der Reverse KL. 14
2.8 Visualisierung der Auswirkung verschiedener Werte für $\alpha$. . . . . 16

3.1 Grundlegende Einteilung der Ansätze in hierarchisch zusammenhängenden Gruppen. . . . . 17
3.2 Veranschaulichung des Prinzips von Mixture-Density-Models. . . . . 26
3.3 Vergleich zwischen Laplace- und Gauß-Verteilung. . . . . 28
3.4 Beispiel einer Sematischen Segmentierung mit getrennter Schätzung für Daten- bzw. Modellunsicherheit. . . . . 29
3.5 Vergleich der Kalibrierung früherer Modelle mit heutigen. . . . . 31
3.6 Aufbau eines Kalibrierungs-Plots. . . . . 32
3.7 Beispiel des zu berechnenden Intervalls. . . . . 36
3.8 Verlauf der Kalibrierungskurve bei Fehlern in der Unsicherheit oder der Prädiktion. . . . . 39
3.9 Beispielhafte Darstellung der Verschiebung der Verteilung der Fehler bei fehlerhafter Schätzung des Mittelwertes. . . . . 41
3.10 Verlauf der Log-Likelihood-Funktionen für Regression und Klassifikation 42

4.1 Histogramm der Trainings-, Validierungs- und Testdaten für den varianzhomogenen Datensatz. . . . . 48
4.2 Streudiagramme der beiden synthetischen Datensätze. . . . . 49
4.3 Anteil der Anomalien pro Ausgabekanal. . . . . 51
4.4 Visualisierung der Annahme an die Anomalien für die Detektion mit Datenunsicherheit . . . . . 54
4.5 Visualisierung der Aufteilung in normale Daten und Anomalien. . . . . 55
4.6 Visualisierung der Annahme an die Anomalien für die Detektion mit Datenunsicherheit . . . . . 56
4.7 Aufteilung der Datensätze pro Ausgabekanal . . . . . 58

5.1 Schema der Modellarchitektur für die synthetischen Daten. . . . . 62
5.2 Verlauf des Mean Squared Error (MSE) in Abhängigkeit der Modellgröße. 63
5.3 Verlauf des MSE in Abhängigkeit der Modellgröße für No-U-Turn Sampler (NUTS) . . . . . 65
5.4 Verlauf der Änderung der Schätzungen des Mittelwertes, der Standardabweichung und der Gauß Log-Likelihood in Abhängigkeit der Anzahl an Vorwärtsprozessen . . . . . 67
5.5 Unterschied der Fehler bei unterschiedlicher Schätzung des Mittelwertes 68
5.6 Verlauf des MSE in Abhängigkeit der Modellgröße für die Mittelwertberechnung anhand der MC-Samples . . . . . 69
5.7 Verlauf des Fehlers in Abhängigkeit der Datengröße . . . . . 70
5.8 Verlauf der mittleren Modellunsicherheit in Abhängigkeit der Modellgröße 74
5.9 Mittleren Modellunsicherheit in Abhängigkeit der Dropout-Rate . . . . 76
5.10 Gelernte Funktionen der Modelle . . . . . 77
5.11 Balkendiagramm der mittleren Modellunsicherheiten für die verschiedenen Daten-Bereiche mit N-Modellen . . . . . 80
5.12 Balkendiagramm der mittleren Modellunsicherheiten für die verschiedenen Daten-Bereiche mit K-Modellen . . . . . 81
5.13 ROC-Kurven für K-Modelle auf varianzheterogenen Daten und Dropout-Rate 0.5. . . . . 83
5.14 Balkendiagramm der AUC-Scores in Abhängigkeit der Dropout-Rate . 84
5.15 Mittlere Datenunsicherheit in Abhängigkeit der Modellgröße. . . . . 86

5.16 Verlauf des MSE der Datenunsicherheit in Abhängigkeit der Modellgröße 87
5.17 MSE der Schätzung der Datenunsicherheit. . . . . 88
5.18 Kalibrierungskurven der K-Modelle. . . . . 89
5.19 MSE der Kalibrierungskurven. . . . . 90
5.20 Verlauf des MSE in Abhängigkeit der Modellgröße . . . . . 93
5.21 MSE pro Ausgabe . . . . . 94
5.22 Vergleich der KPI-Werte für Ausgabekanal 0 und 1. . . . . 96
5.23 Verlauf des Fehlers für Anomaliedetektion anhand der Modellunsicherheit 98
5.24 Verlauf der Modellunsicherheit in Abhängigkeit der Modellgröße . . . . 100
5.25 Verlauf der Datenunsicherheit in Abhängigkeit der Modellgröße . . . . 101
5.26 Kalibierungskurven für alle Ausgabekanäle . . . . . 103
5.27 ROC-Kurven für jeden Ausgabekanal auf Basis der Datenunsicherheit . 105
5.28 ROC-Kurven für jeden Ausgabekanal auf Basis der Modellunsicherheit 109
5.29 Darstellung eines Worst Case für die Anomaliedetektion anhand der Unsicherheiten . . . . . 112

A.1 Beispiel einer Lernkurve . . . . . 117
A.2 Vergleich der Aktivierungsfunktionen . . . . . 119
A.3 Entwicklung der Model-Unsicherheit in Abhängigkeit der Sample-Anzahl. 120
A.4 Beispiel Optimierung der Dropout-Rate . . . . . 122
A.5 Verlauf des MSE in Abhängigkeit der Minibatch-Größe . . . . . 123
A.6 Datensatztests für verschiedene Dropout-Raten . . . . . 124

# Tabellenverzeichnis

4.1 Beschreibung der Eingabebereiche. . . . . 47
4.2 Angaben zur Datensatzgröße . . . . . 48

5.1 Tabelle der verwendeten Hyperparameter beim Training auf synthetischen Daten. . . . . 60
5.2 Tabelle der verwendeten Hyperparameter beim Training auf Daten der STEAG Energy Services GmbH (SES). . . . . 92

# Liste der Algorithmen

3.1 Pseudocode für den generellen Ablauf von MCMC-Methoden. . . . . 24
3.2 Pseudocode zum allgemeinen Ablauf des Verfahrens aus [Osband et al., 2018]. Dieser Pseudocode stammt dabei direkt aus der Arbeit und wurde lediglich übersetzt. . . . . 30
3.3 Pseudocode zur Berechnung der Kalibrierungskurve für die Klassifikation. 34
3.4 Pseudocode für die Berechnung der Kalibrierungskurve für die Regression. 37

# Literaturverzeichnis

[Amari, 1985] Amari, Shun'ichi (1985). *Differential-Geometrical Methods in Statistic*. Springer. *(Seite: 14, 15)*

[Bishop, 1994] Bishop, Christopher M. (1994). *Mixture Density Networks*. Technischer Bericht, Citeseer. *(Seite: 25, 26)*

[Blei et al., 2016] Blei, David M., A. Kucukelbir und J. D. McAuliffe (2016). *Variational Inference: A Review for Statisticians*. Journal of the American statistical association (JASASS), 112(518):859–877. *(Seite: 21, 24)*

[Chen et al., 2018] Chen, Yutian, A. Huang, Z. Wang, I. Antonoglou, J. Schrittwieser, D. Silver und N. de Freitas (2018). *Bayesian Optimization in AlphaGo*. CoRR, abs/1812.06855. *(Seite: 2)*

[Frazao und Alexandre, 2014] Frazao, Xavier und L. A. Alexandre (2014). *Dropall: Generalization of two convolutional neural network regularization methods*. In: *International Conference Image Analysis and Recognition (ICIAR)*, S. 282–289. Springer. *(Seite: 8)*

[Gal, 2016] Gal, Yarin (2016). *Uncertainty in Deep Learning*. Doktorarbeit, University of Cambridge. *(Seite: 21)*

[Gal und Ghahramani, 2016a] Gal, Yarin und Z. Ghahramani (2016a). *Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference*. In: *4th International Conference on Learning Representations workshop track (ICLR)*, International Conference on Learning Representations. *(Seite: 22)*

[Gal und Ghahramani, 2016b] Gal, Yarin und Z. Ghahramani (2016b). *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning*.

In: *Proceedings of the 33rd International Conference on Machine Learning (ICML). (Seite: 21, 22, 27, 40, 44, 60, 66, 67, 68, 72, 82, 108, 113, 114)*

[Gal et al., 2017] Gal, Yarin, J. Hron und A. Kendall (2017). *Concrete Dropout.* In: *Advances in Neural Information Processing Systems (NeurIPS)*, Neural Information Processing Systems. *(Seite: 22, 114, 121)*

[Gelfand und Smith, 1990] Gelfand, Alan E und A. F. Smith (1990). *Sampling-based approaches to calculating marginal densities.* Journal of the American statistical association (JASASS), 85(410):398–409. *(Seite: 23)*

[Gelman et al., 1992] Gelman, Andrew, D. B. Rubin et al. (1992). *Inference from iterative simulation using multiple sequences.* Statistical science, 7(4):457–472. *(Seite: 24)*

[Guo et al., 2017] Guo, Chuan, G. Pleiss, Y. Sun und K. Q. Weinberger (2017). *On Calibration of Modern Neural Networks.* In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*, International Conference on Machine Learning. *(Seite: 30, 31, 34, 40)*

[Hastings, 1970] Hastings, W Keith (1970). *Monte Carlo sampling methods using Markov chains and their applications. (Seite: 23)*

[He et al., 2016] He, Kaiming, X. Zhang, S. Ren und J. Sun (2016). *Deep residual learning for image recognition.* In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, S. 770–778. *(Seite: 122)*

[Hernández-Lobato et al., 2016] Hernández-Lobato, José Miguel, Y. Li, M. Rowland, D. Hernández-Lobato, T. D. Bui und R. E. Turner (2016). *Black-Box Alpha-Divergence Minimization.* In: *Proceedings of the 33rd International Converence on Machine Learning (ICML)*, International Conference on Machine Learning. *(Seite: 15, 16)*

[Hoffman und Gelman, 2014] Hoffman, Matthew D. und A. Gelman (2014). *The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo.* Journal of Machine Learning Research (JMLR), 45(1):1593–1623. *(Seite: 24, 25, 44, 64, 72, 113)*

[Hron et al., 2017] Hron, Jiri, A. G. d. G. Matthews und Z. Ghahramani (2017). *Variational Gaussian Dropout is not Bayesian*. CoRR, arXiv:1711.02989. *(Seite: 23)*

[Huang et al., 2016] Huang, Gao, Y. Sun, Z. Liu, D. Sedra und K. Q. Weinberger (2016). *Deep networks with stochastic depth*. In: *European Conference on Computer Vision (ECCV)*, S. 646–661. Springer. *(Seite: 8)*

[Jordan et al., 1999] Jordan, Michael I, Z. Ghahramani, T. S. Jaakkola und L. K. Saul (1999). *An introduction to variational methods for graphical models*. Machine learning (ML), 37(2):183–233. *(Seite: 21)*

[Kendall und Gal, 2017] Kendall, Alex und Y. Gal (2017). *What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?*. In: *Advances in Neural Information Processing Systems (NeurIPS)*, Neural Information Processing Systems. *(Seite: 27, 29, 31, 44, 62, 82, 85, 91, 108, 113, 114)*

[Kingma et al., 2015] Kingma, Diederik P., T. Salimans und M. Welling (2015). *Variational Dropout and the Local Reparameterization Trick*. In: *Advances in Neural Information Processing Systems (NeurIPS)*, Neural Information Processing Systems. *(Seite: 23)*

[Kiureghian und Ditlevsen, 2009] Kiureghian, Armen Der und O. Ditlevsen (2009). *Aleatory or epistemic? Does it matter?*. Structural Safety, 31(2):105–112. *(Seite: 5)*

[Kullback und Leibler, 1951] Kullback, Solomon und R. A. Leibler (1951). *On information and sufficiency*. Annals of Mathematical Statistics (AOMS), 22(1):79–86. *(Seite: 11)*

[Lakshminarayanan et al., 2017] Lakshminarayanan, Balaji, A. Pritzel und C. Blundell (2017). *Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles*. In: *Advances in Neural Information Processing Systems (NeurIPS)*, Neural Information Processing Systems. *(Seite: 28)*

[Li, 2018] Li, Yingzhen (2018). *Approximate Inference: New Visions*. Doktorarbeit, University of Cambridge. *(Seite: 21)*

[Li und Gal, 2017] Li, Yingzhen und Y. Gal (2017). *Dropout Inference in Bayesian Neural Networks with Alpha-divergences*. In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*, International Conference on Machine Learning. *(Seite: 22)*

[Liu und Wang, 2016] Liu, Qiang und D. Wang (2016). *Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm*. In: *Advances in Neural Information Processing Systems (NeurIPS)*, Neural Information Processing Systems. *(Seite: 19, 25)*

[Minka, 2001] Minka, Thomas (2001). *Expectation Propagation for Approximate Baysian Inference*. In: *Proceedings of the 17th conference on Uncertainty in artificial intelligence (UAI)*, Conference on Uncertainty in artificial intelligence, S. 362–369. *(Seite: 21)*

[Minka, 2004] Minka, Thomas (2004). *Power EP*. *(Seite: 21)*

[Nair und Hinton, 2010] Nair, Vinod und G. E. Hinton (2010). *Rectified linear units improve restricted boltzmann machines*. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*, S. 807–814. *(Seite: 60, 92)*

[Neal, 2011] Neal, Radford M. (2011). *MCMC using Hamiltonian dynamics*. Handbook of Markov Chain Monte Carlo, 2(11). *(Seite: 24, 25)*

[Neal, 2012] Neal, Radford M. (2012). *Bayesian learning for neural networks*. Doktorarbeit. *(Seite: 21, 24)*

[Neumann et al., 2018] Neumann, Lukas, A. Zisserman und A. Vedaldi (2018). *Relaxed Softmax: Efficient Confidence Auto-Calibration for Safe Pedestrian Detection*. *(Seite: 31)*

[Osband et al., 2018] Osband, Ian, J. Aslanides und A. Cassirer (2018). *Randomized Prior Functions for Deep Reinforcement Learning*. In: *Advances in Neural Information Processing Systems (NeurIPS)*, S. 8626–8638. *(Seite: 22, 28, 30, 75, 139)*

[Srivastava et al., 2014] Srivastava, Nitish, G. Hinton, A. Krizhevsky, S. Ilya und R. Salakhutdinov (2014). *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. Journal of Machine Learning Research (JMLR), 15(1):1929–1958. *(Seite: 7, 8, 68, 69)*

[van Erven und Harremoës, 2014] van Erven, Tim und P. Harremoës (2014). *Rényi Divergence and Kullback-Leibler Divergence*. IEEE Transactions on Information Theory (TIT), 60(7):3797–3820. *(Seite: 15)*

[Wald, 1945] Wald, Abraham (1945). *Sequential tests of statistical hypotheses*. Annals of Mathematical Statistics (AOMS), 16(2):117–186. *(Seite: 51)*

[Wan et al., 2013] Wan, Li, M. Zeiler, S. Zhang, Y. Le Cun und R. Fergus (2013). *Regularization of neural networks using dropconnect*. In: *International Conference on Machine Learning (ICML)*, S. 1058–1066. *(Seite: 8)*

[Zeiler, 2012] Zeiler, Matthew D (2012). *ADADELTA: an adaptive learning rate method*. CoRR, arXiv:1212.5701. *(Seite: 60, 92)*

[Zhang et al., 2017] Zhang, Cheng, J. Butepage, H. Kjellstrom und S. Mandt (2017). *Advances in Variational Inference*. CoRR, arXiv:1711.05597. *(Seite: 21)*

[Zhu und Rohwer, 1995] Zhu, Huaiyu und R. Rohwer (1995). *Information geometric measurements of generalisation*. Technischer Bericht, Aston University. *(Seite: 15)*