Fortschritt-Berichte VDI

VDI

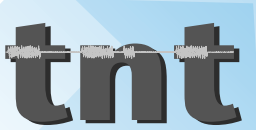
Reihe 10

Informatik/
Kommunikation

Dipl.-Ing. Thorsten Laude,
Langenhagen

Nr. 871

# Konturbasierte multidirektionale Intra-Prädiktion für die Videocodierung




Institut für Informationsverarbeitung
www.tnt.uni-hannover.de

# Konturbasierte multidirektionale Intra-Prädiktion für die Videocodierung

Von der Fakultät für Elektrotechnik und Informatik

der Gottfried Wilhelm Leibniz Universität Hannover

zur Erlangung des akademischen Grades

**Doktor-Ingenieur**

genehmigte

**Dissertation**

von

**Dipl.-Ing. Thorsten Laude**

geboren am 29. Dezember 1988 in Hannover

2020

| | |
|---|---|
| Hauptreferent: | Prof. Dr.-Ing. Jörn Ostermann |
| Korreferent: | Prof. Dr.-Ing. Jens-Rainer Ohm |
| Vorsitzender: | Prof. Dr.-Ing. Bodo Rosenhahn |
| Tag der Promotion: | 6. November 2020 |

Fortschritt-Berichte VDI

Reihe 10

Informatik/
Kommunikation

Dipl.-Ing. Thorsten Laude,
Langenhagen

Nr. 871

# Konturbasierte multidirektionale Intra-Prädiktion für die Videocodierung

**Institut für Informationsverarbeitung**
www.tnt.uni-hannover.de

Laude, Thorsten
**Konturbasierte multidirektionale Intra-Prädiktion für die Videocodierung**
Fortschr.-Ber. VDI Reihe 10 Nr. 871. Düsseldorf: VDI Verlag 2021.
160 Seiten, 36 Bilder, 12 Tabellen.
ISBN 978-3-18-387110-0, ISSN 0178-9627,
€ 57,00/VDI-Mitgliederpreis € 51,30.

**Keywords:** Videocodierung – HEVC – Intra-Prädiktion – maschinelles Lernen – Gauß-Prozesse – Deep Learning – neuronale Netzwerke – stochastische Prozesse

In dieser Arbeit werden zwei Verfahren zur Verbesserung der Intra-Prädiktion für die Videocodierung vorgeschlagen. Der erste Beitrag in dieser Arbeit besteht aus einem stochastischen Konturmodell zur Modellierung und Extrapolation von Konturen, die im Referenzbereich detektiert werden. Für die Modellierung wird ein Gauß-Prozess verwendet. Für die Konturextrapolation wird eine multivariate Gauß-Verteilung formuliert. Der zweite Beitrag in dieser Arbeit ist ein auf neuronalen Netzwerken basierendes Verfahren zur Abtastwertprädiktion. Mit den neuronalen Netzwerken werden die benachbarten Referenzabtastwerte sowie das Ergebnis der Konturmodellierung und -extrapolation als Eingabedaten verarbeitet, um eine Prädiktion der Abtastwerte des zu codierenden Blocks zu erzeugen. Die Codierungseffizienz des Videocodecs HEVC wird um bis zu 5% gesteigert.

**Bibliographische Information der Deutschen Bibliothek**
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliographie; detaillierte bibliographische Daten sind im Internet unter www.dnb.de abrufbar.

**Bibliographic information published by the Deutsche Bibliothek**
(German National Library)
The Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliographie (German National Bibliography); detailed bibliographic data is available via Internet at www.dnb.de.



Als Manuskript gedruckt. Printed in Germany.

ISSN 0178-9627

ISBN 978-3-18-387110-0

# VORWORT

Diese Dissertation habe ich als wissenschaftlicher Mitarbeiter des Instituts für Informationsverarbeitung (TNT) der Gottfried Wilhelm Leibniz Universität Hannover geschrieben. Meinem Doktorvater Professor Jörn Ostermann danke ich für die exzellente Betreuung, seine Ideen und Anregungen, das Hauptreferat zu dieser Dissertation, stets beste Arbeitsbedingungen und alles, was ich wissenschaftlich und andersweitig gelernt habe. Schon während meines Studiums gab er mir die Gelegenheit, am TNT zu forschen und an internationalen Standardisierungsaktivitäten teilzunehmen.

Professor Bodo Rosenhahn danke ich für zahlreiche wissenschaftliche Diskussionen und dafür, dass er als Vorsitzender der Prüfungskommission den reibungslosen Ablauf der Promotion unter den Bedingungen der Corona-Pandemie ermöglicht hat. Professor Jens-Rainer Ohm danke ich für die Übernahme des Korreferats.

Meinen ehemaligen Arbeitskolleginnen und -kollegen, von denen viele inzwischen enge Freunde sind, danke ich für ihre Hilfsbereitschaft, ihren Rat und für viele schöne gemeinsame Erlebnisse. Hierzu zählen unter anderem Hendrik Hachmann, der darüber hinaus ein ausgezeichneter Trauzeuge war, Issi Zell, Felix Kuhnke, Bastian Wandt, Stella Graßhof, Jan Voges, Marco Munderloh, der während meiner Zeit am TNT mein Mentor war, Holger Meuel, Benjamin Spitschan, Aron Sommer, Matthias Reso und Matthias Schuh, der das Institut technisch am Laufen hält und stets für eine gute Atmosphäre sorgt. Sontje Ihler danke ich ebenfalls für ihre Freundschaft und zahlreiche Diskussionen. Wegen ihnen war die Zeit am TNT eine sehr gute und schöne Zeit.

Matthias Narrosche danke ich, dass er mich zur Videocodierung gebracht hat. Solveig Behr, Doris Jaspers-Göring, Pia Bank, Hilke Brodersen, Thomas Wehberg und Martin Pahl danke ich für die kompetente administrative und technische Unterstützung.

Den Mitgliedern des Arbeitssaals Dachkammer danke ich für die unvergessliche Zeit und die gemeinsame Unterstützung während des Studiums. Hierzu zählen unter anderem Niklas Briest, Moritz Wallat, Malte John sowie Felix Burghardt.

Meiner Frau Jennifer Laude danke ich für ihre Liebe, ihre unermüdliche Unterstützung und auch für ihr Verständnis für lange Arbeitszeiten während der Fertigstellung der Dissertation. Meinen Eltern Birgit und Burkhard Laude danke ich dafür, dass sie mich während meines gesamten

Werdegangs unterstützt haben. Ohne sie wäre diese Arbeit nicht möglich gewesen. Meinen Schwiegereltern Petra und Norbert Lübke danke ich für die Aufnahme in ihre Familie.

# INHALTSVERZEICHNIS


1 EINLEITUNG 1
- 1.1 Motivation . . . 1
- 1.2 Stand der Forschung . . . 4
- 1.3 Ungelöste Probleme . . . 6
- 1.4 Ziele der Arbeit . . . 7
- 1.5 Aufbau der Arbeit . . . 9

2 GRUNDLAGEN 10
- 2.1 Videocodierung . . . 10
- 2.2 Maschinelles Lernen . . . 27
- 2.3 Konturdetektion . . . 49

3 VERFAHREN ZUR MODELLIERUNG VON KONTUREN 53
- 3.1 Konturdetektion . . . 53
- 3.2 Konturglättung . . . 56
- 3.3 Modellierung der Kontur . . . 57
- 3.4 A-Priori-Gauß-Prozess . . . 58
- 3.5 Posterior-Gauß-Prozess . . . 59
- 3.6 Konturextrapolation . . . 63
- 3.7 Einbettung in das Gesamtsystem . . . 65

4 VERFAHREN ZUR ABTASTWERTPRÄDIKTION MITTELS MASCHINELLEN LERNENS 67
- 4.1 Einordnung in das Gesamtsystem . . . 67
- 4.2 Datenbasis . . . 67
- 4.3 Architekturen . . . 73
- 4.4 Training . . . 78

5 EXPERIMENTELLE UNTERSUCHUNG UND BEWERTUNG 84
- 5.1 Integration in einen Bildcodec . . . 85
  - 5.1.1 Mehrwert des vorgeschlagenen Konturmodells . . . 89
  - 5.1.2 Mehrwert der vorgeschlagenen Abtastwertprädiktion 93
  - 5.1.3 Einordnung von Effizienz und Komplexität . . . 96
- 5.2 Integration in einen Videocodec . . . 101
  - 5.2.1 Polynomielles Konturmodell und Abtastwertprädiktion mit neuronalen Netzwerken . . . 109
  - 5.2.2 Stochastisches Konturmodell und Abtastwertprädiktion mit neuronalen Netzwerken . . . 112

6 ZUSAMMENFASSUNG 119

LITERATUR 123

# FORMELZEICHEN

## VIDEOCODIERUNG

| | |
|---|---|
| $r$ | Datenrate |
| $H$ | Entropie |
| $S_{\text{horizontal}}$ | Horizontale Chroma-Unterabtastung |
| $I$ | Informationsgehalt |
| $\Xi\Phi\Psi$ | Konstanten zur Modellierung der Prädiktionsfehlerkoeffizientendatenrate |
| $\lambda$ | Lagrange-Faktor für die Rate-Distortion-Kosten |
| $S_Y$ | Luma-Abtastrate |
| $S_{\text{Max}}$ | Maximaler Signalwert |
| $\mathcal{C}$ | Prädiktionsfehlerkoeffizienten |
| $R$ | Rate |
| $C_{RD}$ | Rate-Distortion-Kosten |
| $S_{\text{vertikal}}$ | Vertikale Chroma-Unterabtastung |
| $D$ | Verzerrung, engl. Distortion |

## KONTURMODELLIERUNG

| | |
|---|---|
| $\Theta_0$ | Anfangswerte der Hyperparameter der Kovarianzfunktion |
| $\mathbf{b}$ | Beobachtungen des zu modellierenden Prozesses („Ground Truth“) |
| $\mathbf{f}$ | Beobachtungen für rauschbehaftete Trainingspunkte |
| $\bar{\mathbf{f}}$ | Beobachtungen für rauschfreie Trainingspunkte |
| $\mathbf{f}_*$ | Beobachtungen für Testpunkte |
| $I$ | Bild, engl. Image |

| | |
|---|---|
| $s$ | Blockgröße |
| $\lvert ... \rvert$ | Determinantenoperator |
| $\mathcal{I}$ | Einheitsmatrix |
| $\mathcal{GP}$ | Gauß-Prozess |
| $\Phi$ | Gradientenrichtungen für die Konturdetektion |
| $\Theta$ | Hyperparameter der Kovarianzfunktion |
| $c_{x_p}$ | Konfidenzfunktion im extrapolierten Konturpunkt $x_p$ |
| $G_{\text{skel}}$ | Konturenskelett |
| $\boldsymbol{K}$ | Konturbild |
| $k(x_p, x_q)$ | Kovarianzfunktion |
| $K(A, B)$ | Kovarianzmatrix für die Zufallsvariablen $A$ und $B$ |
| $\Sigma$ | Kovarianzmatrix inklusive unkorreliertem weißen Rauschen |
| $\xi$ | Längsskalierungsfaktor für $k_{\text{SE}}(x_p, x_q)$ |
| $\mathcal{N}$ | Normal-Verteilung |
| $\widetilde{\mathbf{b}}$ | Prior-Beobachtungen für den modellierten Prozess (Prädiktion) |
| $\mathbf{p}$ | Prädiktionen (Posterior-Beobachtungen) |
| $K_{\text{post}}$ | Posterior-Kovarianzmatrix |
| $\Sigma_{\text{post}}$ | Posterior-Kovarianzmatrix mit Rauschen |
| $\zeta$ | Skalierungsfaktor für $k_{\text{SE}}(x_p, x_q)$ |
| $K_{\text{Sobel}}$ | Sobel-Operator |
| $k_{\text{SE}}(x_p, x_q)$ | *Squared Exponential*-Kernel bzw. -Kovarianzfunktion |
| $\sigma_{pi}$ | Standardabweichung der Prädiktion für den Punkt $x_{pi}$ |
| $x_{*i}, \mathbf{x}_*$ | Testpunkte |
| $T$ | Tiefpassfilter für die Konturdetektion |
| $x_i, \mathbf{x}$ | Trainingspunkte |

| | |
|---|---|
| $\epsilon$ | unkorrelliertes weißes Rauschen |
| $L$ | untere Dreiecksmatrix nach Cholesky-Zerlegung |
| $\sigma_{pi}^2$ | Varianz der Prädiktion für den Punkt $x_{pi}$ |
| $\sigma_n^2$ | Varianz des unkorrelierten weißen Rauschens $\epsilon$ |
| $p(...)$ | Wahrscheinlichkeitsdichte für das Argument |
| $P(...)$ | Wahrscheinlichkeit für das Argument |
| $x_{pi}, \mathbf{x}_p$ | x-Werte der Posterior-Prädiktionen |
| $y_i, \mathbf{y}$ | y-Werte der Konturpixel |
| $y_{pi}, \mathbf{x}_p$ | y-Werte der Posterior-Beobachtungen |

## MASCHINELLES LERNEN

| | |
|---|---|
| $a$ | Aktivierungsfunktion |
| $o$ | Aktivierungssignal eines Neurons |
| $\alpha$ | Aktualisierungsschrittweise beim Gradientenabstiegsverfahren mit Momentum |
| $N$ | Anzahl an verwendeten Trainingsbeispielen |
| $\boldsymbol{b}$ | Biasvektor einer Schicht eines neuronalen Netzwerks |
| $i$ | Eingangssignal eines Neurons |
| $e$ | Eulersche Zahl |
| $e_{\text{gen}}$ | Generalisierungsfehler |
| $w$ | Gewicht eines neuronalen Netzwerks |
| $\mathbf{W}$ | Gewichtsmatrix einer Netzwerkschicht |
| $\mathbf{W_F}$ | Gewichtsmatrix einer Faltungsschicht |
| $b$ | Gleichanteil, engl. bias, eines Neurons |
| $\nabla_c C$ | Gradient von $C$ nach $\boldsymbol{c}$ |
| $g_i$ | Gradient zum Zeitpunk $i$ |

| | |
|---|---|
| $\beta_1, \beta_2$ | Hyperparameter des Adam-Verfahrens |
| $l$ | Index der Schicht eines neuronalen Netzwerkes |
| $\mathcal{O}()$ | Komplexität |
| $\boldsymbol{c}$ | Konfiguration eines neuronalen Netzwerks |
| $C$ | Kostenfunktion für das Training eines neuronalen Netzwerks |
| $v_i$ | Leistung zum Zeitpunkt $i$ |
| $\eta$ | Lernrate |
| $N^*$ | Minibatch-Größe |
| $m_i$ | Mittelwert zum Zeitpunkt $i$ |
| $\mu$ | Momentum |
| $e_{\text{Test}}$ | Testfehler |
| $e_{\text{Training}}$ | Trainingsfehler |
| $o(i)$ | Übertragungsfunktion eines Neurons |

# AKRONYME

| | |
|---|---|
| Adam | Adaptive Moment Estimation |
| AI | all intra |
| AV1 | AOMedia Video 1 |
| AVC | Advanced Video Coding |
| BAC | binäre arithmetische Codierung |
| BD | Bjøntegaard Delta |
| Bins | binäre Symbole vor der Entropiecodierung |
| Bits | binäre Symbole nach der Entropiecodierung |
| BVI | Bristol Vision Institute |
| CABAC | Context-adaptive Binary Arithmetic Coding |
| CAVLC | Context-adaptive Variable Length Coding |
| CIE | Commission Internationale de l'Éclairage |
| CoMIC | Contour-based Multidirectional Intra Coding |
| CAE | Convolutional Auto Encoder |
| CNN | Convolutional Neural Network |
| CPU | Central Processing Unit |
| CTB | Coding Tree Block |
| CTC | Common Test Conditions |
| CTU | Coding Tree Unit |
| CU | Coding Unit |
| DC | Direct Current |
| DCT | Discrete Cosine Transform |
| DPB | Decoded Picture Buffer |

| | |
|---|---|
| DPCM | Differential Pulse Code Modulation |
| DST | Discrete Sine Transform |
| GOP | Group of Pictures |
| GPU | Graphics Processing Unit |
| HDF5 | Hierarchical Data Format 5 |
| HEVC | High Efficiency Video Coding |
| HLS | High-level Syntax |
| HM | High Efficiency Video Coding (HEVC) Test Model |
| IBC | Intra Block Copy |
| ILSVRC | ImageNet Large Scale Visual Recognition Challenge |
| ITU-T | International Telecommunication Union - Telecommunication Standardization Sector |
| IEC | International Electrotechnical Commision |
| ISO | International Standardization Organisation |
| JBIG | Joint Bi-level Image Experts Group |
| JCT-VC | Joint Collaborative Team on Video Coding |
| JEM | Joint Exploration Model |
| JPEG | Joint Photographic Experts Group |
| JVET | Joint Video Experts Team |
| KLT | Karhunen-Loève-Transformation |
| KODIM | Kodak Image Dataset |
| lreLU | Leaky Rectifier Linear Unit |
| MIP | Matrix-weighted Intra Prediction |
| MPEG | Moving Picture Experts Group |
| MPM | Most Probable Modes |
| MSE | Mean Squared Error |

| | |
|---|---|
| MV | Motion Vector |
| NAL | Network Abstraction Layer |
| NMS | Non-maximum Suppresion |
| PB | Prediction Block |
| PCA | Principal Component Analysis |
| Pel | picture element |
| PSNR | Peak Signal-to-Noise Ratio |
| PU | Prediction Unit |
| QP | Quantisierungsparameter |
| RAISE | Raw Image Dataset |
| RD | Rate-Distortion |
| RDO | Rate-Distortion Optimization |
| ReLU | Rectifier Linear Unit |
| RGB | Rot-Grün-Blau |
| RNN | Recurrent Neural Network |
| SATD | Sum of Absolute Transformed Differences |
| SSIM | Structural Similarity |
| TB | Transform Block |
| TPU | Tensor Processing Unit |
| TU | Transform Unit |
| USC-SIPI | University of Southern California - Signal and Information Processing Institute |
| VCEG | Video Coding Experts Group |
| VTM | Versatile Test Model |
| WPP | Wavefront Parallel Processing |

## ABSTRACT

The amount of transmitted video data is growing faster than the channel capacity available for this purpose. This leads to the necessity of a continuous improvement of the coding methods for the used video codecs. Modern video codecs are generally based on the principle of hybrid coding, i.e. the combination of a prediction with a transformation coding of the prediction error. The prediction methods can be roughly divided into intra and inter prediction. In this work, two methods are proposed for improving intra prediction.

The first contribution in this thesis is a stochastic contour model for modeling and extrapolation of contours detected in the reference area. A Gaussian process is used for the modeling. Expectations of typically occurring contour shapes were taken into account by choosing the *Squared Exponential Kernel* as covariance function of the prior of the Gaussian process. The posterior Gaussian process resulted from the prior Gaussian process by optimizing the hyperparameters of the covariance function for each contour. A multivariate Gaussian distribution was formulated for the contour extrapolation. The second contribution in this thesis is a neural network-based method for sample value prediction. The neural networks are used to process the adjacent reference sample values and the results of contour modeling and extrapolation as input data to generate a prediction of the sample values of the block to be coded. The contours are required for the sample value prediction. The neural networks were designed with an auto-encoder architecture and trained using a SATD cost function.

The coding efficiency of the video codec HEVC was increased by up to 5%. Averaged over all 55 test sequences, the *All Intra* configuration resulted in BD-rates of $-0.54\%$ for high bit rates and $-1.0\%$ for low bit rates. Compared to the methods from our own prior works, which were already better than related works from the literature, an additional increase in coding efficiency of 0.21 percentage points for high bit rates and 0.27 percentage points for low bit rates was achieved.



***Keywords*** – video coding, HEVC, intra prediction, machine learning, Gaussian process.

# KURZFASSUNG

Die zur Übertragung von Videos benötigte Übertragungskapazität wächst schneller als die hierfür zur Verfügung stehende Kanalkapazität. Hieraus entsteht die Notwendigkeit einer stetigen Verbesserung der Codierungsverfahren für die verwendeten Videocodecs. Moderne Videocodecs beruhen in der Regel auf dem Prinzip der Hybridcodierung, also der Kombination von einer Prädiktion mit einer Transformationscodierung des Prädiktionsfehlers. Die Prädiktionsverfahren können grob in Intra- und Inter-Prädiktion unterschieden werden. Für die Verbesserung der Intra-Prädiktion werden in dieser Arbeit zwei Verfahren vorgeschlagen.

Der erste Beitrag in dieser Arbeit besteht aus einem stochastischen Konturmodell zur Modellierung und Extrapolation von Konturen, die im Referenzbereich detektiert werden. Für die Modellierung wird ein Gauß-Prozess verwendet. Die Erwartungen an typischerweise vorkommende Konturverläufe werden durch die Wahl des *Squared Exponential Kernels* als Kovarianzfunktion des Prior-Gauß-Prozesses berücksichtigt. Der Posterior-Gauß-Prozess ergibt sich aus dem Prior-Gauß-Prozess durch die Optimierung der Hyperparameter der Kovarianzfunktion für jede Kontur. Für die Konturextrapolation wird eine multivariate Gauß-Verteilung formuliert. Der zweite Beitrag in dieser Arbeit ist ein auf neuronalen Netzwerken basierendes Verfahren zur Abtastwertprädiktion. Mit den neuronalen Netzwerken werden die benachbarten Referenzabtastwerte sowie das Ergebnis der Konturmodellierung und -extrapolation als Eingabedaten verarbeitet, um eine Prädiktion der Abtastwerte des zu codierenden Blocks zu erzeugen. Die Konturen werden für die Abtastwertprädiktion benötigt. Die neuronalen Netzwerke wurden mit einer Autoencoder-Architektur entworfen und mittels einer SATD-Kostenfunktion trainiert.

Die Codierungseffizienz des Videocodecs HEVC wurde um bis zu 5% gesteigert. Gemittelt über alle 55 Testsequenzen ergaben sich für die *All Intra*-Konfiguration BD-Raten von $-0,54\%$ für hohe Datenraten und in Höhe von $-1,0\%$ für niedrige Datenraten. Verglichen mit den Verfahren aus eigenen Vorarbeiten, welche bereits besser waren als verwandte Arbeiten aus der Literatur, wurde eine zusätzliche Steigerung der Codierungseffizienz um $0,21$ Prozentpunkte für hohe Datenraten und um $0,27$ Prozentpunkte für niedrige Datenraten erzielt.

***Stichworte*** – Videocodierung, HEVC, Intra-Prädiktion, maschinelles Lernen, Gauß-Prozesse

# 1 EINLEITUNG

## 1.1 MOTIVATION

Aus einer Prognose [18] des Telekommunikationsunternehmens Cisco geht hervor, dass die für die Internetübertragung von Videos benötigte Übertragungskapazität rasant ansteigt: Der Prognose zufolge werde hierfür eine Vervierfachung von 2016 bis 2021 auf dann 2,7 Zettabyte pro Jahr[1] (entspräche einer Millionen Minuten Videomaterial pro Sekunde) erwartet. Im gleichen Zeitraum werde lediglich eine Verdoppelung der durchschnittlichen Kanalkapazität von Internetanschlüssen angenommen. Um die Übertragung dieser Daten über die zur Verfügung stehenden Kanäle zu ermöglichen, ist eine stetige Verbesserung von Videocodierungsverfahren erforderlich.

In den vergangenen Jahren konnten hierdurch motiviert enorme Verbesserungen von Videocodierungsverfahren beobachtet werden. Im Januar 2013 finalisierte das Joint Collaborative Team on Video Coding (JCT-VC) – hierbei handelt es sich um eine gemeinsame Standardisierungsgruppe von der International Telecommunication Union - Telecommunication Standardization Sector (ITU-T) Video Coding Experts Group (VCEG) und International Standardization Organisation (ISO)/International Electrotechnical Commision (IEC) Moving Picture Experts Group (MPEG) – die technische Arbeit am neusten Videocodierungsstandard HEVC [47, 116, 119, 140]. Der HEVC Standard wurde von den beiden Standardisierungsgremien als Recommendation H.265 (ITU-T) und als 23008-2:2013 MPEG-H Part 2 (ISO/IEC) veröffentlicht. Bei gleicher visueller Qualität ermöglicht HEVC, abhängig von der gewählten Konfiguration, eine deutliche Reduktion der Datenrate um 40-60% bezogen auf den Vorgängerstandard Advanced Video Coding (AVC), auch bekannt als ITU-T Recommendation H.264 und ISO/IEC MPEG-4 Part 10 [44, 80, 95]. Diese Messwerte wurden sowohl in Experimenten mit den entsprechenden Referenzsoftwareimplementierungen aus der Standardisierung [35, 38, 94] als auch in Experimenten mit realen Produktimplementierungen [20, 34] ermittelt. Im Anschluss an die Finalisierung der ersten Version des HEVC Standards wurden mehrere Erweiterungen (Range Extensions [28],

1 entspräche dann 82% aller über das Internet übertragener Daten

Scalable HEVC [11], HEVC Screen Content Coding [144]) entwickelt. Auch außerhalb der Standardisierungsaktivitäten des JCT-VC entstehen moderne Videocodierungsverfahren, zum Beispiel AOMedia Video 1 (AV1) [16], VP8 [5], VP9 [86] und andere [9, 87, 130]. Einen anderen Ansatz wählen Toderici et al. [125, 126], indem sie Bilder mit rückgekoppelten neuronalen Netzwerken codieren. Hierbei wird das komplette Codierungssystem einschließlich Binarisierung und Entropiecodierung durch Netzwerke realisiert (Ende-zu-Ende). Die Codierungseffizienz der Ende-zu-Ende-Ansätze lässt sich zwischen der von JPEG 2000 und HEVC-Intra einordnen. In neueren Arbeiten ist die Codierungseffizienz weiter gestiegen.

Wie viele erfolgreiche Videocodierungsstandards zuvor beruht auch HEVC auf dem Prinzip der Hybridcodierung, also auf der Kombination von einer Prädiktion (bewegungskompensierend oder intra) mit einer Transformationscodierung des Prädiktionsfehlers [92]. Hierbei wird das Ziel verfolgt, den aktuell zu codierenden Teil des Videosignals aus bereits codierten Signalanteilen zu prädizieren. Da diese bereits codierten Signalanteile am Decoder vorliegen, kann die Prädiktion ebenfalls am Decoder durchgeführt werden. Der bei der Prädiktion entstehende Prädiktionsfehler wird an den Decoder übertragen. Die zur Codierung benötigte Datenrate sinkt je besser die Prädiktion funktioniert. Zu dem Zweck der Prädiktion werden die zu codierenden Bilder des Videos in Blöcke aufgeteilt, welche nacheinander codiert werden. Bei der Codierung eines Blockes stehen die Informationen von bereits codierten Blöcken aus dem aktuellen Bild und aus zuvor codierten Bildern als Prädiktionsgrundlage zur Verfügung.

Die Codierungsverfahren, welche zu der erfolgreichen Verbreitung dieser Videocodierungsstandards geführt haben, können grob in Inter-Codierung und in Intra-Codierung unterschieden werden. Während die Intra-Codierung ausschließlich örtliche Redundanz im jeweiligen zu codierenden Bild ausnutzt, wird bei der Inter-Codierung zusätzlich die zeitliche Redundanz zwischen aufeinanderfolgenden Bildern mittels einer bewegungskompensierenden Prädiktion ausgenutzt [30]. Deshalb werden bei der Verwendung der Intra-Codierung typischerweise deutlich höhere Datenraten (Faktor 10-100 [69]) als bei der Inter-Codierung benötigt, um eine annähernd konstante visuelle Qualität zu erzielen.

Dennoch ist die Intra-Codierung ein essentieller Teil von allen Videocodierungsverfahren und -anwendungen: Sie wird für den Start der Übertragung, für den wahlfreien Zugriff in laufende Übertragungen, zur Fehlerkorrektur (hier teilweise auch in Kombination mit zeitlichen Informationen), für die Umschaltung der Datenrate bei schwankenden Datenkanälen bei Video-on-Demand Anwendungen [20] und nicht zuletzt für die Codierung von neu im Bild erscheinenden Inhalten benötigt.

Des Weiteren sind Intra-Codierungsverfahren prädestiniert für die effiziente Codierung von Einzelbildern. Dieses Szenario kann als Sonderfall der Codierung einer Videosequenz mit nur einem Bild aufgefasst werden.

Die Intra-Codierung in HEVC basiert auf der örtlichen Prädiktion von Abtastwerten[2] auf Grundlage von benachbarten, bereits codierten Abtastwerten (Referenzabtastwerte) gefolgt von einer Transformationscodierung des Prädiktionsfehlers [63]. Für die Prädiktion stehen 33 direktionale Modi, ein Modus für die planare Prädiktion sowie ein Modus für die Prädiktion mit dem Mittelwert der Referenzabtastwerte zur Verfügung. Die Referenzabtastwerte befinden sich in einer 1 Pel breiten Spalte beziehungsweise Zeile innerhalb der bereits codierten Blöcke direkt links und oberhalb des aktuell zu codierenden Blockes (Referenzbereich). Die direktionalen Modi erlauben die lineare Extrapolation der Referenzabtastwerte in der Richtung von 33 verschiedenen Winkeln. Da in typischen Videosequenzen häufig horizontale und vertikale Strukturen vorkommen, wurde hierfür eine genauere Winkelauflösung als für andere Winkel gewählt [82]. Diese direktionalen Modi zielen auf die Prädiktion von Blöcken mit einer linearen Struktur oder Kontur ab [63]. Nicht alle zu codierenden Blöcke enthalten eine lineare Struktur. Für solche Blöcke wurden die Prädiktion mit dem Mittelwert sowie die planare Prädiktion (gewichtete Überlagerung von vier Referenzabtastwerten) entwickelt [63].

Aktuell untersuchen VCEG und MPEG das Potential für die Standardisierung eines HEVC-Nachfolgers im hierzu gegründeten Joint Video Experts Team (JVET), vormals: Joint Video Exploration Team [69, 93]. Im entwickelten Test-Modell, Versatile Test Model (VTM), vormals: Joint Exploration Model (JEM), finden sich mehrere Verbesserungen zur Intra-Codierung: die Anzahl an direktionalen Modi wird auf 65 erhöht, die Interpolation für die direktionalen Modi wird verbessert, zur Verhinderung von Diskontinuitäten an den Blockgrenzen werden prädizierte Randabtastwerte gefiltert, die Chrominanzen werden aus der Luminanz prädiziert und die Filterung von Referenzabtastwerten wird modifiziert [15]. Das Matrix-weighted Intra Prediction (MIP)-Verfahren beruht auf einer Prädiktion mittels einer Matrixmultiplikation. Dieser Ansatz ist vergleichbar mit einem neuronalen Netzwerk mit einer einzigen verborgenen Schicht und ohne Aktivierungsfunktion.

Eine Analyse der im HEVC-Standard und im VTM eingesetzten Intra-Codierung offenbart mehrere Ansätze für Verbesserungen: 1) Die Prädiktionsbasis ist mit 1 Pel Breite beziehungsweise Höhe klein. Es könnten aus weiteren benachbarten und bereits codierten Abtastwerten zusätzliche für die Prädiktion nützliche Informationen gewonnen werden. 2)

2 In dieser Arbeit werden die Begriffe *Bildpunkt* und *Abtastwert* unterschieden. Ein Bildpunkt besteht typischerweise aus drei Abtastwerten für die drei Farbraumkomponenten (z.B. YCbCr).

Die direktionale Prädiktion erlaubt nur eine Prädiktion von linearen Strukturen. 3) Im Fall der direktionalen Prädiktion ist pro Block nur eine Richtung für die Prädiktion wählbar. Schwierig zu prädizieren sind deshalb Blöcke, in denen nichtlineare Konturen oder mehrere Konturen in unterschiedlichen Richtungen vorhanden sind.

Zu einer vergleichbaren Erkenntnis kommen Lottermann und Steinbach in ihrer Arbeit [78], in der sie die Datenrate signalabhängig unter anderem über die örtliche Aktivität modellieren. Die Schlussfolgerung von Lottermann und Steinbach (hohe Datenrate für Blöcke mit vielen Konturen) passt zu der dieser Arbeit zugrunde liegenden Prämisse, dass Blöcke mit vielen Kanten schwierig zu prädizieren sind.

## 1.2 STAND DER FORSCHUNG

Verwandte Arbeiten aus der Literatur zeigen auf, dass sich aus Konturen in der Umgebung des zu codierenden Blockes für die Prädiktion wertvolle Informationen ziehen lassen. Yuan und Sun demonstrieren, dass sich basierend auf einer Konturenkarte für das zu codierende Bild eine schnelle Entscheidungsfindung für die Wahl eines guten Intra-Codierungsmodus entwickeln lässt [146]. Asheri et al. [2] sowie Au und Chan [3] verwenden Konturinformationen für Fehlerverschleierungsalgorithmen. Hierbei werden Konturinformationen aus dem aktuellen Bild (Asheri et al.) beziehungsweise aus dem aktuellen und dem vorherigen Bild (Au und Chan) für die Verschleierung von Übertragungsfehlern eingesetzt. Liu et al. steigern die Codierungseffizienz des Joint Photographic Experts Group (JPEG)-Standards [46, 48, 133] in [76] sowie von JPEG 2000 und AVC in [75, 77] mit einem konturbasierten Inpainting- (Rekonstruktion von gestörten Bildteilen) und Textursyntheseverfahren. Das Grundprinzip dieses Verfahrens beruht auf der linearen Extrapolation von im bereits codierten Bild gefundenen Konturen, der Übertragung von nichtlinearen Konturverläufen mittels des Joint Bi-level Image Experts Group (JBIG)-Standards [45, 49], der Übertragung von Vorlagen für die Textursynthese sowie dem Lösen von partiellen Differentialgleichungen für das Inpainting. Im Gegensatz zum HEVC Intra-Codierungsverfahren wird durch dieses Verfahren die Prädiktion von mehreren Konturen in unterschiedliche Richtungen pro Block ermöglicht.

Liu et al. gelingt eine Steigerung der Codierungseffizienz, jedoch werden nur unkomplizierte lineare Konturen extrapoliert. Des Weiteren müssen für die Abtastwertberechnung entweder zahlreiche Seiteninformationen übertragen werden (Vorlagen für die Textursynthese) oder rechenaufwendige partielle Differentialgleichungssysteme für das Inpainting gelöst werden (die Decodierung eines Bildes mit einer Auflösung von

512x512 Pel benötigt mehrere Minuten [75]). Da die Leistungsfähigkeit nur für ausschließlich intra-codierte, engl. all intra (AI), Bilder demonstriert wird [75, 77] kann nicht angenommen werden, dass die visuell gutaussehenden Ergebnisse der Textursynthese auch eine gute Prädiktion für die Codierung darstellen. Zur Kompensation der beschriebenen Nachteile des Verfahrens von Liu et al. wurde in einer eigenen Vorarbeit das Verfahren Contour-based Multidirectional Intra Coding (CoMIC) für die Verbesserung der Codiereffizienz von JPEG und HEVC vorgeschlagen [65]. Prinzipiell wird hierbei eine JPEG-Codierung um eine Prädiktion ergänzt und die diskrete Kosinustransformation, engl. Discrete Cosine Transform (DCT), auf dem Prädiktionsfehler anstatt auf dem zu codierenden Signal berechnet. Durch die Übertragung des Prädiktionsfehlers wird das Problem von Liu et al., dass die synthetisierten Bildbereiche nicht als Prädiktionsgrundlage geeignet sind, umgangen. CoMIC, hier in der ersten Version (CoMIC v1), basiert wie das Verfahren von Liu et al. auf der getrennten Verarbeitung von Konturen und der Abtastwertprädiktion. Die Konturen werden im bereits codierten Bildbereich detektiert und ohne die Übertragung von Seiteninformationen linear in den zu codierenden Block extrapoliert. Die Abtastwertprädiktion beruht auf der Fortführung von benachbarten Abtastwerten entlang der extrapolierten Konturen. Da die Konturmodellierung dieses Verfahrens in weit von den Blockgrenzen zu den bekannten Bildbereichen entfernten Bereichen des Blockes eine zu hohe Unsicherheit über den Konturverlauf aufweist, erfolgt eine distanzabhängige Überblendung des fortgesetzten Randabtastwertes zum Mittelwert der Referenzabtastwerte. Im Gegensatz zu der im HEVC Intra-Codierungsverfahren eingesetzten Abtastwertprädiktion werden die einzelnen Konturen individuell behandelt, wodurch die Abtastwerte in mehreren Richtungen pro Block fortgeführt werden können. Dieses Verfahren eignet sich für die Prädiktion von linearen Konturverläufen. In realen Bildern und Videosequenzen existieren viele nichtlineare Konturen, welche mit CoMIC v1 nicht prädiziert werden können. Zur Prädiktion dieser nichtlinearen Konturen wurden in einer Nachfolgearbeit (CoMIC v2) die Konturen mit Polynomen höheren Grades modelliert [69]. Durch die in den eigenen Vorarbeiten vorgeschlagenen Verfahren konnte die Codierungseffizienz von HEVC gesteigert werden.

Die extrapolierten Konturen werden für die Abtastwertprädiktion benötigt. Ein ursprünglich aus dem Bereich der Klassifikation bekannter Trend, der seit einigen Jahren auch in der Videocodierung präsent ist und für die Abtastwertprädiktion vielversprechend erscheint, ist Deep Learning [31, 71]. Hierbei handelt es sich um eine Art von maschinellen Lernverfahren, bei der neuronale Netzwerke mit vielen aufeinanderfolgenden Schichten zur Verarbeitung der Eingangsdaten eingesetzt werden.

Häufig werden neuronale Faltungsnetzwerke, engl. Convolutional Neural Networks (CNNs), verwendet. Für Klassifikationsanwendungen werden mit Deep Learning-Ansätzen seit einigen Jahren Ergebnisse erzielt, die dem bisherigen Stand der Technik deutlich überlegen sind [61, 122] und eine rege Aktivität zur Verbesserung der entsprechenden Lernverfahren motivieren [50, 113]. Aufgrund des großen Erfolges von Deep Learning-Ansätzen in anderen Bereichen werden diese auch im Bereich der Bildverarbeitung vermehrt angewendet. Fawzi et al. [27] und Gupta et al. [36] verwenden neuronale Netzwerke für das Inpainting von Bildern. Theis und Bethge [124] sowie Gregor et al. [33] verwenden rückgekoppelte neuronale Netzwerke für die generative Bildmodellierung.

## 1.3 UNGELÖSTE PROBLEME

Die noch nicht hinreichend gelösten Aufgaben bei der Intra-Codierung sind die folgenden:

**Modellierung von Konturen im bereits codierten Signal:** Es konnte in der Literatur und in den eigenen Vorarbeiten gezeigt werden, dass durch die Prädiktion von Abtastwerten entlang von Konturen die Codierungseffizienz deutlich gesteigert werden kann. Jedoch erlaubt keines der beschriebenen Verfahren die allgemeingültige Modellierung von Konturen. Die Modellierungen aus der eigenen Vorarbeit [65] sowie aus den Arbeiten von Liu et al. [75–77] eignen sich ihrer linearen Form entsprechend nicht für nichtlineare Konturen. Des Weiteren konnte gezeigt werden, dass die nichtlinearen Modelle aus [69] jeweils nur für einen kleinen Anteil der in typischen Bildsignalen vorkommenden nichtlinearen Konturen geeignet sind. Dieses ist intuitiv erklärbar, da es zweifelhaft ist, dass von der Natur geschaffene Formen genau die Form von quadratischen oder kubischen Funktionen haben. Selbst bei der Betrachtung des Inhaltes von einzelnen Blöcken mit typischen Blockgrößen reicht eine solche polynominelle Modellierung nicht für die zuverlässige Approximation der tatsächlichen Konturverläufe. In der Vorarbeit [129] wurde gezeigt, dass das polynomielle Konturmodell besser als Splines funktioniert. Für die zuverlässige Konturmodellierung wird ein Modell benötigt, welches zu den realen Daten passt.

**Extrapolation von Konturen:** Neben der Modellierung von Konturen im bereits codierten Bereich des Bildes ist für die Codierung eine Extrapolation in den zu codierenden Block notwendig. Das Verfahren von Liu et al. benötigt für komplizierte, zum Beispiel nichtlineare, Konturen die Übertragung des Konturverlaufes [75–77]. Das CoMIC v1 Verfahren aus der eigenen Vorarbeit ermöglicht eine zuverlässige Prädiktion nur in den Bereichen des zu codierenden Blockes, welche in der Nähe des Referenz-

bereiches liegen [65]. Die unterschiedlichen nichtlinearen Extrapolationsverfahren aus der Nachfolgearbeit ermöglichen die Extrapolation jeweils nur für einen kleinen Teil der vorhandenen Konturen [69].

**Abtastwertprädiktion:** Keines der beschriebenen Verfahren ermöglicht eine zufriedenstellende Prädiktion der Abtastwerte. Die Anforderungen für ein zufriedenstellendes Verfahren sind, dass die Prädiktion genau ist, dass die Prädiktion aus in bereits codierten Bildbereichen vorhandenen Informationen erfolgt, dass keine zusätzlichen Seiteninformationen übertragen werden und dass die Komplexität des Prädiktionsverfahrens gering genug ist, um eine Anwendung in Decodern zu ermöglichen. Die vorgestellten Verfahren verfehlen diese Anforderungen. So ist die Prädiktion der Verfahren aus den Vorarbeiten beim Vorliegen von komplizierten Konturverläufen nicht hinreichend genau genug. In der Vorarbeit [24] wurde gezeigt, dass neuronale Netzwerke alleine noch keine hinreichend gute Intra-Prädiktion ermöglichen.

In den vier auf neuronalen Netzwerken basierenden Arbeiten [27, 33, 36, 124] ist das Ziel die Rekonstruktion eines visuell überzeugenden Bildes. Solch eine visuell überzeugende Rekonstruktion ist jedoch nicht notwendigerweise eine für eine effiziente Codierung geeignete Prädiktion, da sie nicht den zu übertragenden Prädiktionsfehler minimiert. Die Ende-zu-Ende-Verfahren von Toderici et al. [125, 126] sind ebenfalls ungeeignet, da sie ein komplettes Codierungssystem umfassen. In vielen Fällen ist stattdessen eine Verbesserung der über Jahrzehnte optimierten Codierungssysteme für Bilder und Videos erwünscht, um existierende Implementierungen, Architekturen und Infrastrukturen weiterverwenden beziehungsweise verbessern zu können.

## 1.4 ZIELE DER ARBEIT

In dieser Arbeit wird das Ziel verfolgt, eine effizientere Intra-Codierung für die Bild- und Videocodierung durch die Kombination von konventionellen Videocodierungsverfahren mit Konturextrapolationsverfahren und maschinellen Lernansätzen für die Abtastwertprädiktion zu entwickeln. Hierfür wird ein Codierungsverfahren entworfen, welches sowohl die genaue Extrapolation von komplizierten (zum Beispiel nichtlinearen) Konturen ermöglicht als auch aufbauend auf den extrapolierten Konturen eine genaue Prädiktion von Abtastwerten erlaubt. Im Einzelnen werden die folgenden Teilziele verfolgt:

Es wird ein Verfahren zur Modellierung von Konturverläufen im bereits codierten Signal entwickelt. Hierbei werden einige gewünschte Eigenschaften des zu entwickelnden Verfahrens besonders berücksichtigt. Das Verfahren ermöglicht eine allgemeingültige Modellierung für

unterschiedliche Arten von real existierenden Konturverläufen. Hierfür reicht ein einzelnes Modell aus anstatt viele verschiedene Modelle für unterschiedliche Arten von Konturverläufen zu benötigen. Dieses Modell basiert auf Gauß-Prozessen.

Aufbauend auf diesem Modell wird die Extrapolation der im bereits codierten Signal gefundenen und modellierten Konturverläufe in den aktuell zu codierenden Bereich realisiert. Die Parameter für die Extrapolation werden aus der Modellierung des bereits codierten Signals bestimmt. Hierdurch wird die Datenrate für die extrapolierte Kontur stark reduziert, da diese Informationen dem Decoder bereits zur Verfügung stehen und somit nicht übertragen werden müssen.

Die extrapolierten Konturverläufe sind lediglich ein Zwischenschritt für die angestrebte Intra-Codierung. Das eigentliche Ziel der Codierung ist die Rekonstruktion der Abtastwerte des zu codierenden Blockes. Deshalb werden diese Abtastwerte prädiziert. Hierfür werden neuronale Netzwerke verwendet.

Die entwickelten Algorithmen sind kompatibel zu bestehenden Codierungssystemen, so dass sie zum Beispiel als zusätzlicher Prädiktionsmode in ein bestehendes Codierungssystem integriert werden können und hierbei bestehende Verfahren zur Transformationscodierung[3] des Prädiktionsfehlers, zur Entropiecodierung, zur Partitionierung der Bilder mit einem Blockraster, zur bewegungskompensierenden Prädiktion, etc. weiterverwendet werden können. Dieses wäre mit Ende-zu-Ende-Ansätzen, welche das komplette Codierungssystem durch eine Blackbox in Form eines neuronalen Netzwerkes ersetzen, unmöglich. Des Weiteren wird hierdurch die Integration in bestehende Verfahren zur Rate-Distortion-Optimierung durch Encoder ermöglicht.

Die Ergebnisse aus den ersten drei Teilzielen werden verwendet, um das entwickelte Verfahren in einen Bild- (CoMIC) und in einen Videocodec (HEVC) zu integrieren. Diese Integration ist für einen fundierten Vergleich mit dem Stand der Technik erforderlich.

Im Rahmen dieses Arbeit sollen unterschiedliche Verfahren für die Konturextrapolation (ein modellbasiertes Verfahren) und für die Abtastwertprädiktion (ein datenbasiertes Verfahren) verwendet werden. Der Grund hierfür liegt in der Erkenntnis aus eigenen Vorarbeiten, dass die Prädiktion mit einem datenbasierten Verfahren nur aus den Abtastwerten des Referenzbereiches keine zufrieden stellenden Ergebnisse liefert, weil Konturverläufe nicht hinreichend berücksichtigt werden. Die Arbeiten von Liu et al. [75–77] bestätigen diese Erkenntnis.

3 Eine Ausnahme bildet die *Mode-dependent Transform*, welche für ein zusätzliches Prädiktionsverfahren zu erweitern wäre.

Zusammengefasst sind die Beiträge in dieser Arbeit:

- Ein auf Gauß-Prozessen basierendes Verfahren zur Modellierung und Extrapolation von Konturen
- Ein Verfahren zur Prädiktion von Abtastwerten mittels neuronaler Netzwerke
- Die Kombination der beiden vorgenannten Verfahren zu einem Intra-Codierungsverfahren
- Die Integration in einen Bild- und in einen Videocodec

Teile der dieser Arbeit zugrunde liegenden Ideen wurden in [65] und [69] veröffentlicht.

## 1.5 AUFBAU DER ARBEIT

Die weitere Arbeit ist wie folgt gegliedert: In Kapitel 2 werden die notwendigen Grundlagen für diese Arbeit beschrieben. Hierzu zählen die HEVC-Videocodierung mit Schwerpunkt auf der Intra-Codierung, maschinelles Lernen mit neuronalen Netzwerken und das verwendete Verfahren zur Konturdetektion. Das vorgeschlagene stochastische Konturmodell wird in Kapitel 3 hergeleitet. Hierfür werden Konturen als gedächtnisbehaftete Quelle analysiert, die Konturen modelliert sowie die Extrapolation der Konturen in den zu codierenden Block formuliert. Aufbauend auf den Ergebnissen des dritten Kapitels wird die vorgeschlagene Abtastwertprädiktion in Kapitel 4 ausgeführt. Im Rahmen dieser Ausführungen werden sowohl die vorgeschlagenen Netzwerkarchitekturen als auch das Training der Netzwerke beschrieben. Die experimentelle Untersuchung und Bewertung der vorgeschlagenen Verfahren wird in Kapitel 5 geschildert. In Kapitel 6 wird diese Arbeit mit einer Zusammenfassung abgeschlossen.

# 2

# GRUNDLAGEN

In diesem Kapitel werden die zum Verständnis dieser Arbeit notwendigen Grundlagen ausgeführt. Die Grundlagen der Videocodierung werden mit Blick auf den HEVC-Standard dargestellt. Maschinelles Lernen wird mit einem Schwerpunkt auf den in dieser Arbeit verwendeten neuronalen Netzwerken inklusive der zugehörigen Lernverfahren eingeführt. Da das in Kapitel 3 vorgeschlagene Modell auf Konturen beruht, werden die verwendeten Konturdetektionsverfahren vorgestellt.

## 2.1 VIDEOCODIERUNG

In diesem Abschnitt werden die Beschreibung von digitalen Videosignalen, die Hybridcodierung, HEVC sowie das Thema Encodersteuerung eingeführt.

### *Beschreibung von Videosignalen*

Videosignale sind die Eingabedaten von Videocodecs. Folglich ist die Beschreibbarkeit von Videosignalen eine notwendige Voraussetzung für die Entwicklung von Videocodierungsverfahren. Im Rahmen dieser Arbeit werden ausschließlich digitale Videosignale betrachtet.

Typische Quellen für Videosignale sind Kameraaufnahmen von natürlichen Inhalten oder die Generierung von künstlichen Inhalten wie Animationen im Computer. Weitere mögliche Quellen umfassen die Digitalisierung von analogen Filmen oder die Übertragung von Bildschirminhalten, beispielsweise via Chromecast oder AirPlay. Mischungen von Videosignalen aus verschiedenen Quellen sind möglich und je nach Anwendungsgebiet sogar üblich. So werden zum Beispiel für Filmproduktionen typischerweise Kameraaufnahmen von natürlichen Inhalten mit im Computer erzeugten künstlichen Inhalten kombiniert. Je nach Quelle unterliegt das Videosignal bereits zu diesem Zeitpunkt der Verarbeitungskette Verzerrungen, beispielsweise durch die Kameralinse.

Im Folgenden wird ohne Einschränkung davon ausgegangen, dass das Videosignal mit einer Kamera aufgenommen wird. In diesem Fall entsteht das Videosignal aus sichtbarem Licht, also aus elektromagnetischen

Wellen mit Wellenlängen zwischen 380nm und 780nm. Die Erzeugung von Videosignalen erfolgt bildweise, d.h. ein Bild wird nach dem anderen aufgenommen. Im Kamerasensor werden die einfallenden elektromagnetischen Wellen zeitlich und örtlich abgetastet sowie quantisiert. Hierdurch entstehen quantisierte Abtastwerte[1].

Die Abtastwerte werden in zweidimensionalen Feldern[2] zu einem Bild angeordnet. Bei einem Grauwertbild besteht das Bild aus einem Feld. Bei Farbbildern besteht das Bild in der Regel aus drei Feldern. Die Abtastwerte an einer örtlichen Position – also ein Wert für Grauwertbilder und drei Werte für Farbbilder – werden als Bildpunkte (engl. pixel oder picture element (Pel)) bezeichnet.

Die zeitliche Abtastung kann über die Bildrate, also die Anzahl an Bildern pro Sekunde, gemessen mit der Einheit Hertz (Hz), beschrieben werden. Die örtliche Abtastung kann über die Bildauflösung beschrieben werden. Die Bildauflösung gibt die Anzahl an Bildpunkten innerhalb eines Bildes in horizontaler und in vertikaler Richtung an. Die Quantisierung[3] kann über die Auflösung des Quantisierers, bei gleichförmigen Quantisierern parametrisiert über die Bittiefe, beschrieben werden.

Die menschliche visuelle Wahrnehmung kann physiologisch in Helligkeitswahrnehmung und Farbwahrnehmung unterschieden werden. Für die Beschreibung von Farbe im Kontext der Videocodierung erfolgt oft eine Bezugnahme auf das Normfarbsystem mit drei Grundfarben der Internationalen Beleuchtungskommision (fr. Commission Internationale de l'Éclairage (CIE)) von 1931. In diesem System wird jede Farbe als ein Punkt in einem XYZ-Raum dargestellt. Die Koordinaten des Punktes werden basierend auf dem einfallenden Lichtspektrum berechnet.

Für Farb-Videosignale operieren die Quelle und die Senke oft im Rot-Grün-Blau (RGB)-Farbraum. Zwischen Quelle und Senke – also für die Codierung – wird das Signal in der Regel in einen anderen Farbraum umgewandelt. Hierfür wird typischerweise der YCbCr-Farbraum[4] verwendet. Hier steht Y für die Helligkeit oder Luma gemäß des CIE-Normfarbsystems und Cb und Cr für die Farbinformationen als Differenzsignale (Cb für die blau-gelbe Chroma und Cr für die rot-grüne Chroma). Für die Aufteilung des Videosignals in einen Helligkeitsanteil und zwei Farbanteile gibt es im Wesentlichen zwei Gründe, von denen

1 Im Folgenden wird davon ausgegangen, dass Abtastwerte immer quantisiert sind.

2 Nicht zu verwechseln mit den Feldern von Videosignalen im Zeilensprungverfahren (engl. interlaced), welche in dieser Arbeit nicht betrachtet werden.

3 Hier ist die Quantisierung während der Analog-zu-Digital-Wandlung und nicht die Quantisierung während der Videocodierung gemeint.

4 In HEVC-Erweiterungen wird teilweise auch der YCgCo-Farbraum verwendet, da hierdurch die Korrelation zwischen den beiden Chroma-Komponenten zusätzlich reduziert werden kann [150].

einer nur noch unter historischen Gesichtspunkten relevant ist: Zu Zeiten des analogen Schwarz-Weiß-Fernsehens gab es nur die Helligkeitskomponente. Als das analoge Farbfernsehen aufkam, wurde die Farbe in Form von zwei Differenzsignalen übertragen. Hierdurch funktionierten alte Schwarz-Weiß-Fernseher, welche nur die Helligkeit anzeigen konnten, ohne Probleme weiter und Farbfernseher, welche zusätzlich die beiden Farbsignale anzeigen konnten, stellten ein farbiges Bild dar [140]. Heute noch relevant ist die unterschiedliche Sensibilität des menschlichen Beobachters für Helligkeits- und Farbunterschiede. Hierbei ist letztere geringer. Durch die Trennung von Helligkeits- und Farbinformationen können die für den menschlichen Beobachter weniger relevanten Farbinformationen mit geringerer Qualität codiert werden, was einen positiven Einfluss auf die benötigte Datenrate hat.

Für Anwendungen mit hohen Qualitätsanforderungen wie zum Beispiel die Filmproduktion oder die Verarbeitung medizinischer Daten werden die Daten nach der beschriebenen Farbraumumwandlung direkt verwendet. Für Anwendungen mit geringeren Qualitätsanforderungen wie zum Beispiel die Ausstrahlung von Filmen für Endkunden lässt sich zunächst die genannte Eigenschaft der menschlichen Wahrnehmung mit einer Farbunterabtastung gezielt ausnutzen um die Datenrate zu senken. Bei der Farbunterabtastung wird die örtliche Auflösung von einer oder beiden Farbkomponenten reduziert. Als Namensschema für unterschiedliche Arten der Farbunterabtastung wird das folgende Verhältnis verwendet:

$$S_\mathrm{Y} : S_\mathrm{horizontal} : S_\mathrm{vertikal}. \tag{2.1}$$

$S_\mathrm{Y}$ steht hierbei für die Luma-Abtastrate, den Bezugwert für die beiden Chroma-Komponenten, und wird aus historischen Gründen und ohne faktische Notwendigkeit auf Vier gesetzt[5]. $S_\mathrm{horizontal}$ kennzeichnet die horizontale Farbabtastung mit Bezug auf die Lumaabtastung. Häufig wird dieser Faktor auf Zwei gesetzt, was einer Halbierung der Auflösung entspricht. $S_\mathrm{vertikal}$ beschreibt die vertikale Farbunterabtastung in Abhängigkeit der horizontalen Farbunterabtastung. Wird dieser Wert auf Null gesetzt, so wird für die vertikale Farbunterabtastung der gleiche Faktor wie für die horizontale Farbunterabtastung gewählt. Wird der Wert auf Zwei gesetzt, so erfolgt keine vertikale Farbunterabtastung. Die gebräuchlichste Farbunterabtastung ist 4:2:0. Erfolgt keine Farbunterabtastung, so wird die Bezeichnung 4:4:4 verwendet. Die unterschiedlichen örtlichen Auflösungen je nach gewählter Farbunterabtastung werden in Abbildung 2.1 visualisiert.

5 Für das analoge Fernsehen wurde die Vierfache NTSC-Videobandbreite für die Abtastung der Helligkeit verwendet.

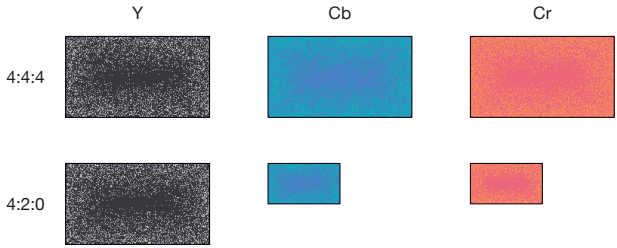



Abbildung 2.1: Schematische Darstellung der Farbunterabtastung für die Fälle 4:4:4 und 4:2:0

Videos können als Sequenz von Einzelbildern betrachtet werden. Die Bilder werden zeitlich hintereinander angeordnet und wiedergegeben. Ist die Bildrate hoch genug, dann nimmt der menschliche Betrachter eine flüssige Bewegung zwischen sich verändernden Bildern wahr. Ab 24Hz, der oft im Kino eingesetzten Bildrate, werden Bewegungen als flüssig wahrgenommen. Für andere Anwendungen werden 50Hz, 60Hz, 120Hz oder mehr eingesetzt.

Die typische Senke für Videosignale war lange Zeit exklusiv der menschliche Betrachter mittels einer Anzeige (z.B. Fernseher, Computerbildschirm, Smartphone, Tablet). Seit einiger Zeit kommen Algorithmen als Senken für Videosignale hinzu (z.B. für die automatische Auswertung von Überwachungsvideos oder für die Inhaltserkennung in hochgeladenen Videos).

In Tabelle 2.1 werden die unkomprimierten Datenraten von Videosignalen mit typischen örtlichen und zeitlichen Auflösungen, Quantisierer-Bittiefen und Farbunterabtastungen aufgelistet. Bei verfügbaren Übertragungsraten in der Größenordnung von einigen Dutzend bis wenigen hundert MBit/s und Speicherkapazitäten von wenigen Terabyte ist ersichtlich, dass hochauflösende Videosignale ohne zusätzliche Codierung nicht sinnvoll übertragen oder gespeichert werden können. Zur Lösung dieses Problems werden Videocodierungsverfahren eingesetzt. Im nachfolgenden Abschnitt wird das Grundprinzip für viele Videocodierungsverfahren eingeführt – die Hybridcodierung.

*Hybridcodierung*

Die Hybridcodierung[6] beschreibt im Kontext der Videocodierung die Kombination einer Prädiktion mit einer Transformationscodierung zur

6 Hybrid: lat. Mischung, zusammengesetzt aus zwei oder mehr Komponenten

Tabelle 2.1: Übersicht über typische Videoformate

| Name | Auflösung | Bittiefe | Farbunter-abtastung | MBit/s (unkomprimiert) |
|---|---|---|---|---|
| UHD 8K | 7680×4320@120Hz | 10 Bit | 4:2:0 | 59 720 |
| | 7680×4320@60Hz | 10 Bit | 4:2:0 | 29 860 |
| | 7680×4320@50Hz | 10 Bit | 4:2:0 | 24 884 |
| | 7680×4320@24Hz | 10 Bit | 4:2:0 | 11 944 |
| | 7680×4320@120Hz | 10 Bit | 4:4:4 | 119 440 |
| | 7680×4320@60Hz | 10 Bit | 4:4:4 | 59 720 |
| | 7680×4320@50Hz | 10 Bit | 4:4:4 | 49 767 |
| | 7680×4320@24Hz | 10 Bit | 4:4:4 | 23 888 |
| UHD 4K | 3840×2160@120Hz | 10 Bit | 4:2:0 | 14 930 |
| | 3840×2160@60Hz | 10 Bit | 4:2:0 | 7 465 |
| | 3840×2160@50Hz | 10 Bit | 4:2:0 | 6 221 |
| | 3840×2160@24Hz | 10 Bit | 4:2:0 | 2 986 |
| | 3840×2160@120Hz | 10 Bit | 4:4:4 | 29 860 |
| | 3840×2160@60Hz | 10 Bit | 4:4:4 | 14 930 |
| | 3840×2160@50Hz | 10 Bit | 4:4:4 | 12 442 |
| | 3840×2160@24Hz | 10 Bit | 4:4:4 | 5 972 |
| 1080p | 1920×1080@60Hz | 8 Bit | 4:2:0 | 1 493 |
| | 1920×1080@50Hz | 8 Bit | 4:2:0 | 1 245 |
| | 1920×1080@30Hz | 8 Bit | 4:2:0 | 747 |
| | 1920×1080@25Hz | 8 Bit | 4:2:0 | 623 |
| 720p | 1280×720@60Hz | 8 Bit | 4:2:0 | 664 |
| | 1280×720@50Hz | 8 Bit | 4:2:0 | 553 |
| | 1280×720@30Hz | 8 Bit | 4:2:0 | 332 |
| | 1280×720@25Hz | 8 Bit | 4:2:0 | 277 |
| WVGA | 832×480@60Hz | 8 Bit | 4:2:0 | 288 |
| | 832×480@50Hz | 8 Bit | 4:2:0 | 240 |
| | 832×480@30Hz | 8 Bit | 4:2:0 | 144 |
| | 832×480@25Hz | 8 Bit | 4:2:0 | 120 |
| WQVGA | 416×240@60Hz | 8 Bit | 4:2:0 | 72 |
| | 416×240@50Hz | 8 Bit | 4:2:0 | 60 |
| | 416×240@30Hz | 8 Bit | 4:2:0 | 36 |
| | 416×240@25Hz | 8 Bit | 4:2:0 | 30 |

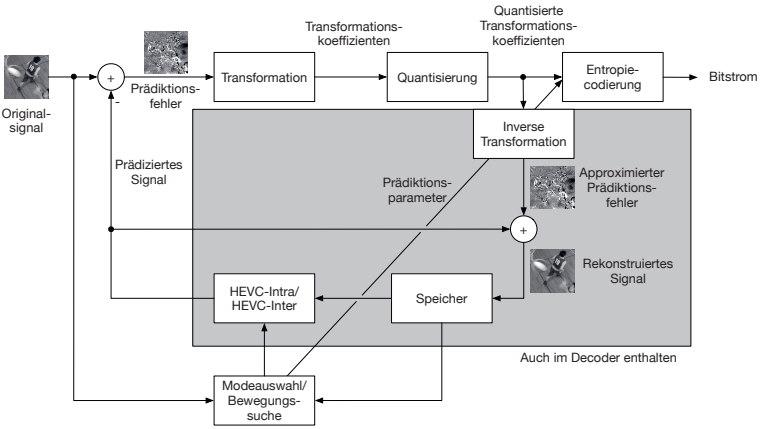



Abbildung 2.2: Blockschaltbild für einen auf der Hybridcodierung aufbauenden Videocodec.

Übertragung des Prädiktionsfehlers. Auch wenn mit der Zeit die einzelnen Teil-Algorithmen in hybriden Videocodierungsverfahren stetig verbessert und erweitert wurden, so ist das Grundprinzip seit dem 1988 veröffentlichten Videocodierungsstandard H.261 unverändert. Im Folgenden wird die Hybridcodierung zunächst aus Sicht eines Videoencoders der Abbildung 2.2 folgend beschrieben[7].

Es erfolgt eine blockweise Codierung des Videosignals in einer Differential Pulse Code Modulation (DPCM)-Schleife. Hierfür wird das aktuell zu codierende Bild des Videos in nicht-überlappende Blöcke aufgeteilt, welche nacheinander verarbeitet werden. Das Ziel der Prädiktion ist die Reduktion von örtlicher und zeitlicher Redundanz im zu codierenden Videosignal. Die Ziele der Transformation sind zum einen die Reduktion von örtlicher Redundanz durch eine Transformation des Prädiktionsfehlers sowie zum anderen die Entfernung der am wenigsten relevanten[8] Informationen aus dem Videosignal durch Quantisierung.

Die Funktionsweise der Prädiktion beruht auf der Vorhersage des aktuellen Blocks aus bereits am Decoder vorhandenen Signalen. Der Vorteil

7 HEVC und andere Videocodierungsstandards spezifizieren in der Regel nur die Syntax und Semantik des Bitstroms sowie den zugehörigen Decodierungsprozess, mit dem aus dem Bitstrom ein Videosignal rekonstruiert werden kann. Die Erzeugung des Bitstroms am Encoder ist nicht standardisiert und liegt im Ermessen des Encoderentwicklers. Zwei verschiedene Encoder für den gleichen Standard können sehr unterschiedliche Ergebnisse produzieren. Zur besseren Lesbarkeit werden in dieser Arbeit dennoch viele Prozesse aus Sicht des Encoders beschrieben, auch wenn diese Sicht streng genommen außerhalb der Standards liegt.

8 Der Begriff *Irrelevanzreduktion* wird hier bewusst nicht verwendet, da je nach Quantisiererstufenanzahl auch (viele) relevante Informationen aus dem Signal entfernt werden.

dieser Funktionsweise ist, dass alle aus am Decoder bereits vorhandenen Informationen prädizierten Signale nicht mehr übertragen werden müssen. Hierdurch hat die Prädiktion einen wesentlichen Anteil an der Reduktion der Datenrate während der Codierung eines Videos. Für die Prädiktion werden bereits decodierte Signale für einen gewissen Zeitraum am Decoder gespeichert, um diese Signale als Referenz für die Prädiktion verwenden zu können.

Die in modernen Videocodecs verwendeten Prädiktionsverfahren können in Intra-Prädiktion und Inter-Prädiktion unterschieden werden.

Bei der Intra-Prädiktion erfolgt eine Prädiktion des aktuellen Blocks aus bereits decodierten Signalen des aktuellen Bildes. Die zugrunde liegende Annahme ist hierbei, dass es örtliche Abhängigkeiten zwischen benachbarten Abtastwerten gibt. Als Referenz werden entweder direkt benachbarte Abtastwerte verwendet (klassische Intra-Prädiktion [63, 95, 139]) oder örtlich weiter entfernte Abtastwerte (zum Beispiel Intra Block Copy (IBC) [145]). Am Encoder findet eine Auswahl des am besten geeigneten Intra-Prädiktionsverfahrens für den aktuellen Block statt. Die Intra-Prädiktion wird unter anderem für die Prädiktion von neu in der Szene erscheinenden Inhalten, für die Codierung von Einzelbildern, für den Start und für den wahlfreien Zugriff bei Übertragungen, für die Fehlerkorrektur und für die Umschaltung der Datenrate bei schwankenden Kanalkapazitäten benötigt. Bei der Intra-Prädiktion entstehen Fehler, wenn das Signal des aktuellen Blockes gegeben der Referenzwerte nicht vollständig mit dem verwendeten Intra-Prädiktionsverfahren vorhergesagt werden kann. Abgesehen von Ausnahmen, wie zum Beispiel sehr homogenen Blöcken, entsteht in der Regel ein nicht zu vernachlässigender Fehler.

Bei der Inter-Codierung erfolgt die Prädiktion des aktuellen Blocks aus dem Signal in bereits vollständig decodierten Bildern, den sogenannten Referenzbildern[9]. Die hierbei getroffene Annahme ist, dass es eine hohe zeitliche Redundanz zwischen zeitlich benachbarten Bildern gibt. Es wird deshalb angenommen, dass es nur wenige Änderungen zwischen diesen Bildern gibt, welche hauptsächlich auf Bewegungen innerhalb der aufgenommenen Szene zurückzuführen sind. Diese Bewegungen werden mit einem Bewegungsmodell beschrieben und kompensiert. Das verwendete Verfahren wird als Bewegungskompensation bezeichnet. Aus einem Ausschnitt des bewegungskompensierten Referenzbilds ergibt sich eine gute Prädiktion für den aktuell codierten Block. In modernen Videocodecs werden zum Beispiel translatorische und affine Bewegungs-

9 Da die Bilder eines Videos nicht in der gleichen Reihenfolge codiert werden müssen, in der sie angezeigt werden, handelt es sich hierbei nicht notwendigerweise um zeitlich vorhergehende Bilder [107].

modelle für die Bewegungskompensation verwendet. Am Encoder findet eine Bewegungssuche statt[10]. Mit dieser Bewegungssuche werden die Parameter für das verwendete Bewegungsmodell, also zum Beispiel der Bewegungsvektor [74] (engl. Motion Vector (MV)) für das translatorische Bewegungsmodell, bestimmt. Inter-Prädiktion wird dann eingesetzt, wenn das Signal des zu codierenden Blocks zumindest näherungsweise in wenigstens einem der Referenzbilder vorhanden ist und durch das Bewegungsmodell kompensiert werden kann. Fehler entstehen bei der Inter-Prädiktion zum Beispiel, wenn die Bewegung durch das verwendete Bewegungsmodell nicht erfasst werden kann, wenn Bildinhalte im aktuellen Bild neu erscheinen oder wenn die Blockpartitionierung nicht perfekt zu den Objektgrenzen in der aufgenommenen Szene passt [58].

Durch Subtraktion des prädizierten Signals von dem Originalsignal wird der Prädiktionsfehler erzeugt. Dieser ist für einen Großteil der Gesamtdatenrate verantwortlich, insbesondere bei mittleren und hohen Datenraten[11]. Benachbarte Werte des Prädiktionsfehlers sind miteinander korreliert. Für eine effiziente Entropiecodierung wäre eine Codierung der Verbundereignisse erforderlich, was in der Praxis nicht sinnvoll umsetzbar ist.

Die Prädiktionsfehlercodierung erfolgt deshalb in zwei Stufen: Zunächst wird der Prädiktionsfehler in einen anderen Raum transformiert. Anschließend werden die Transformationskoeffizienten quantisiert. Hierbei werden die folgenden beiden Ziele verfolgt: Erstens wird eine Dekorrelation der Koeffizienten angestrebt. Wenn dieses Ziel erreicht wird, dann ist die Summe der Einzelentropien der Koeffizienten gleich der Entropie des Verbundsereignisses aller Koeffizienten. Dann können die einzelnen Koeffizienten unabhängig voneinander codiert werden anstatt der Codierung des Verbundereignisses ohne dass die Codierungseffizienz sinkt. Zweitens soll der Prädiktionsfehler mit möglichst wenig Transformationskoeffizienten beschrieben werden (Energiekompaktheit).

Die Karhunen-Loève-Transformation (KLT)[12] ist die beste Transformation für das Erreichen der beiden genannten Ziele. Jedoch konnte sich die KLT als datenabhängige Transformation in der Praxis nicht durchsetzen, da für jeden zu codierenden Prädiktionsfehler eine eigene Transformationsmatrix berechnet und übertragen werden müsste. Da die Transformationsmatrix für einen Block der Größe $N \times N$ die Größe $N^2 \times N^2$ hat, ist die Menge an Seiteninformationen zu groß.

10 Es gibt auch Ansätze für eine Bewegungssuche am Decoder zur Verbesserung der Inter-Prädiktion [17, 59].

11 Klomp misst 75% für den oberen Bereich an Rundfunkqualität [57].

12 Die KLT ist sehr eng mit der Hauptkomponentenanalyse, engl. Principal Component Analysis (PCA), verwandt. Sie unterscheiden sich je nach Definition nur um eine Normierung.

Stattdessen werden häufig DCT-Varianten[13] verwendet. Für den konstruierten Fall, dass die Korrelationsmatrix des Prädiktionsfehlers eine Toeplitz-Matrix ist, sind die KLT und DCT asymptotisch äquivalent [109]. Für reale Prädiktionsfehler ist die DCT nicht ganz so gut wie die KLT. Teilweise werden auch Discrete Sine Transform (DST)-Varianten verwendet, da diese für kleine, intra-prädizierte Blöcke besser zu den Werten an den Rändern der Prädiktionsfehlerblöcken passen. Seit AVC werden die Transformationen in Ganzzahl-Arithmetik spezifiziert, um abweichende Ergebnisse zwischen unterschiedlichen Fließkomma-Implementierungen zu vermeiden.

Die Transformationskoeffizienten werden quantisiert. In der Regel werden hierfür skalare, gleichförmige Quantisierer verwendet. Unter der Annahme, dass die Koeffizienten Laplace-verteilt sind, werden die Grenzen zwischen den Quantisiererstufen nicht mittig sondern verschoben zwischen den Repräsentativwerten angeordent, um den Quantisierungsfehler zu minimieren. Optional kann eine Skalierung der Koeffizienten erfolgen. Hierbei wird die Matrix der Prädiktionsfehlerwerte elementweise durch eine Skalierungsmatrix geteilt. Häufig wird bei der Skalierung berücksichtigt, dass die menschliche visuelle Wahrnehmung sensibler für niedrige als für hohe Frequenzen ist. Deshalb werden die Koeffizienten für die hohen Frequenzen durch größere Zahlen geteilt als die Koeffizienten für niedrige Frequenzen.

Die quantisierten Transformationskoeffizienten werden für die Rekonstruktion des Signals verwendet. Hierfür wird die inverse Transformation auf sie angewendet. Der entstehende approximierte Prädiktionsfehler weicht aufgrund der Quantisierung vom ursprünglichen Prädiktionsfehler ab. Durch die Addition des prädizierten Signals und des approximierten Prädiktionsfehlers entsteht das rekonstruierte Signal. Das rekonstruierte Signal wird in einem Speicher, dem sogenannten Referenzbildspeicher, engl. Decoded Picture Buffer (DPB), gespeichert, um angezeigt zu werden und um als Referenz für zukünftige Prädiktionen verwendet zu werden.

Alle für die Rekonstruktion benötigten Informationen werden an den Decoder übertragen. Dieses umfasst die Information für die Durchführung der Prädiktion, zum Beispiel Bewegungsvektoren oder das ausgewählten Intra-Prädiktionsverfahren, sowie die quantisierten Transformationskoeffizienten. Für die zu übertragenden Informationen wird eine Entropiecodierung angewendet. Das Ziel der Entropiecodierung ist die verlustfreie Codierung der zu übertragenen Informationen, welche gege-

13 Häufig wird die DCT-II als Transformation und die DCT-III als inverse Transformation verwendet. In modernen Codecs werden zusätzlich weitere Varianten der DCT verwendet und adaptiv ausgewählt.

benenfalls bereits verlustbehaftet sind, mit möglichst geringer Datenrate. In den betrachteten Videocodierungsverfahren wird eine binäre arithmetische Codierung (BAC) verwendet. Hierbei werden binäre Symbole codiert[14]. Gegebenenfalls erfolgt eine Binarisierung von nicht-binären Symbolen, zum Beispiel mittels Zählcodes oder Varianten von Golombcodes. Bei der BAC wird eine Folge von Bins als eine Fließkommazahl aus einem berechneten Intervall innerhalb eines definierten Wertebereichs codiert[15]. Hieraus ergibt sich der Vorteil, dass Symbole mit weniger als einem Bit codiert werden können. Dieses ist mit anderen Codes, wie zum Beispiel Huffman-Codes, nicht möglich. Während der Codierung werden die für die Codierung genutzten Symbolwahrscheinlichkeiten an die Statistik der codierten Daten angepasst. Das resultierende Verfahren wird als kontext-adaptive BAC, engl. Context-adaptive Binary Arithmetic Coding (CABAC), bezeichnet [118].

Von den beschriebenen Teilen der Hybridcodierung sind die inverse Transformation zur Erlangung des approximierten Prädiktionsfehlers, die Prädiktion, die Rekonstruktion des Signals sowie der Speicher mit Referenzen für die Prädiktion ebenfalls am Decoder vorhanden.

### *High Efficiency Video Coding*

In diesem Abschnitt werden relevante Aspekte des Videocodecs HEVC [47, 116, 140] ausgeführt.

Im vorherigen Abschnitt wurde für die Schilderung der Hybridcodierung vereinfachend angenommen, dass es bei der Codierung nur eine Art von Blöcken gibt und dass diese Blöcke alle gleich groß sind. Das ist in modernen Videocodecs wie HEVC nicht der Fall. Zudem gibt es bei HEVC die Unterscheidung zwischen *Block* und *Unit*: Ein Block umfasst ein Feld mit Abtastwerten, welche einem rechteckigen Bereich innerhalb eines Bildes zugeordnet sind. Eine Unit umfasst alle Blöcke für diesen Bereich, also drei Blöcke für Farbsignale oder einen Block für monochrome Signale. Die Zuordnungen sind:

- Coding Tree Unit (CTU) - Coding Tree Block (CTB)
- Transform Unit (TU) -Transform Block (TB)
- Prediction Unit (PU) - Prediction Block (PB)

Die CTU bildet die erste Stufe der Blockpartitionierung und damit den Ausgang für alle weiteren Partitionierungen. CTUs sind quadratisch und

14 Der Nomenklatur im Kontext von HEVC folgend werden binäre Symbole vor der Entropiecodierung (Bins) und binäre Symbole nach der Entropiecodierung (Bits) unterschieden.

15 In CABAC weicht die Realisierung hiervon ab.

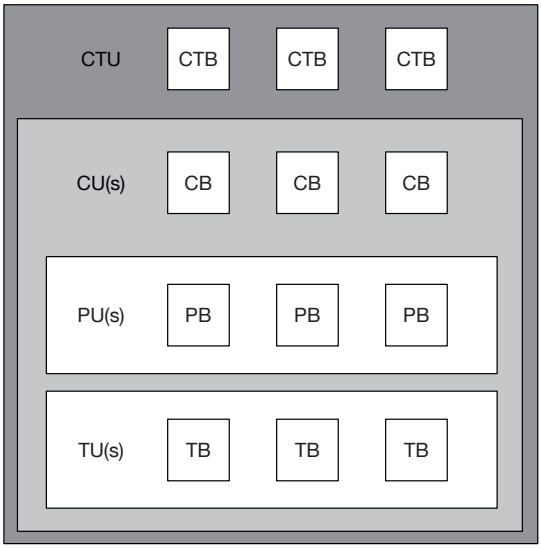



Abbildung 2.3: Übersicht über die in HEVC verwendeten Blockarten

haben eine feste Größe, die im Rahmen der Syntax eines Parametersets auf einen der drei folgenden Werte festgelegt wird: $16 \times 16$, $32 \times 32$ und $64 \times 64$. Für große örtliche Videoauflösungen ist eine große CTU-Größe sinnvoll, für kleine Auflösungen eine kleine CTU-Größe [140]. Die zu codierenden Bilder werden in gleichgroße und nicht überlappende CTUs aufgeteilt. Es erfolgt eine Verarbeitung der CTUs in Rasterscan-Reihenfolge.

Jede CTU bildet die Wurzel für zwei unabhängige Quaternärbäume (engl. Quadtrees) aus Coding Unit (CU)s und TUs. Hierbei ist der TU-Quaternärbaum dem CU-Quaternärbaum nachgelagert.

Auf Ebene der CU wird die Auswahl zwischen Intra- und Inter-Codierung getroffen. Anschließend können die CUs weiter in ein, zwei oder vier PUs aufgeteilt werden.

Die PUs dienen als Ebene für die Durchführung der eigentlichen Prädiktion. Maximal können die PUs so groß sein wie die zugehörigen CUs. Die minimale Größe für die Intra-Prädiktion, welche auf TU-Ebene stattfindet, ist $4 \times 4$, $4 \times 8$ beziehungsweise $8 \times 4$ für die Inter-Prädiktion mit einem Referenzbild sowie $8 \times 8$ für die Inter-Prädiktion mit zwei Referenzbildern. Die Beschränkungen für die Inter-Prädiktion dienen der Komplexitätslimitierung für Decoder. Alle PUs sind rechteckig aber nicht notwendigerweise quadratisch.

Die quadratischen TUs werden für die Transformationscodierung verwendet. Ihre maximale Größe entspricht für die Inter-Codierung der CU-Größe, die TUs können also größer sein als die für die Prädiktion dieses Bildbereiches verwendeten PUs. Dieses ist zum Beispiel bei mit dem Skip Mode codierten Blöcken sinnvoll [140]. Für die Intra-Codierung können die TUs maximal so groß sein wie die zugehörigen PUs, da das rekonstruierte Signal, für das der decodierte Prädiktionsfehler verwendet wird, für die Prädiktion der nächsten PU benötigt wird.

Aufeinanderfolgende CTU eines Bildes in Verarbeitungsreihenfolge (Rasterscan) werden in einem Slice zusammengefasst. Ein Bild kann aus einem oder mehreren Slices bestehen. Somit sind Slices nicht notwendigerweise rechteckig. Slices sind unabhängig voneinander decodierbar – bei einem verlorenen Slice können die verbleibenden Slices eines Bildes codiert werden – und stellen ein Werkzeug zur Fehlereindämmung dar. In HEVC werden drei Arten von Slices unterschieden: I-Slices, in denen die Blöcke ausschließlich mit Intra-Prädiktion codiert werden, P-Slices, in denen die Blöcke mit Intra-Prädiktion oder mit Inter-Prädiktion mit einem Referenzbild codiert werden, und B-Slices, in denen Intra-Prädiktion sowie Inter-Prädiktion mit einem oder mit zwei Referenzbildern eingesetzt werden. Zwischen den Slices eines Bildes bestehen keine Abhängigkeiten bezüglich der Prädiktion, der Bestimmung von Syntaxelementen oder bezüglich der Entropiecodierung. Slices dienen hauptsächlich zur Fehlereindämmung während der Übertragung, jedoch können sie potentiell auch für die Parallelisierung der Encodierung und Decodierung des Videos eingesetzt werden. Allerdings gibt es hierbei zwei nachteilige Auswirkungen [84]: Damit Slices unabhängig decodierbar sind ist es erforderlich, dass Slice Header-Informationen redundant im Bitstrom vorhanden sind. Insbesondere bei niedrigen Datenraten ist diese Redundanz nicht vernachlässigbar[16]. An den Grenzen zwischen Slices kann es zu Blockartefakten kommen, da keine Filterung über Slice-Grenzen hinaus möglich ist.

Tiles stellen eine Möglichkeit zur Parallelisierung der Encodierung und Decodierung ohne die genannten Nachteile von Slices dar. Hierbei wird das Bild entlang des CTU-Rasters in rechteckige Bereiche aufgeteilt. Die Verarbeitungsreihenfolge der CTU eines Bildes wird angepasst: Für jedes Tile gibt es einen unabhängigen Rasterscan. Die Tiles eines Bildes können parallel codiert werden, jedoch sind sie nicht unabhängig voneinander: Falls die zu einem Tile gehörenden Daten verloren gehen, dann können die verbleibenden Tiles möglicherweise nicht fehlerfrei decodiert werden. Da keine Informationen zum Fehlerschutz redundant übertragen werden und da eine Filterung des decodierten Signals über Tile-Grenzen hinweg

16 Auf *dependent slice segments* [108] wird an dieser Stelle nicht näher eingegangen.

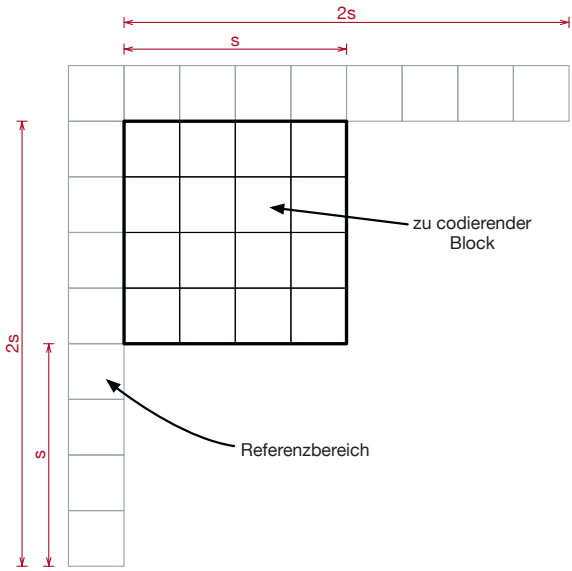



Abbildung 2.4: HEVC-Referenzbereich für die Intra-Prädiktion. In Anlehnung an [63].

zulässig ist, entstehen die genannten Parallelisierungsnachteile von Slices nicht.

Schließlich können mehrere Bilder zu Gruppen, engl. Group of Pictures (GOP), zusammengefasst werden. Für weitere Informationen zur sogenannten High-level Syntax (HLS) von HEVC sei der Leser auf [111] verwiesen.

Die HEVC-Intra-Prädiktion ist das Referenzverfahren, welches durch die Beiträge in dieser Arbeit verbessert werden soll. Deshalb wird sie im Folgenden detailliert geschildert.

Die Abtastwerte innerhalb eines 1 Pel breiten beziehungsweise hohen Streifens in bereits codierten Blöcken an der Grenze zum aktuellen Block wie in Abbildung 2.4 dargestellt dienen als Referenzwerte für die Prädiktion. Für einen $s \times s$-Block gibt es $4s + 1$ Referenzwerte. Aus der Abbildung ist ersichtlich, dass sich Teile des Referenzbereichs in den diagonal links-unten und rechts-oben gelegenen PB befinden. Je nach Partitionierung des Bildes sind diese PB bereits codiert und stehen als Prädiktionsreferenz zur Verfügung oder sie wurden noch nicht codiert und stehen somit nicht zur Verfügung. Wenn Referenzwerte nicht referenzierbar sind, zum Beispiel weil sie noch nicht codiert wurden oder außerhalb des Bildes, Slices oder

Tiles liegen, dann werden sie auf den Wert des örtlich nächsten verfügbaren Referenzwertes gesetzt. Sind gar keine Referenzwerte verfügbar, zum Beispiel für den ersten PB eines Bildes, Slices oder Tiles, dann werden alle Referenzwerte auf den mittleren Wert des Wertebereichs der Abtastwerte, also 128 für 8 Bit Signale, gesetzt. Abhängig von der Blockgröße und des Intra-Prädiktionsmodus werden die Referenzabtastwerte vor der Prädiktion mit einem Tiefpassfilter geglättet[17].

Für die Intra-Prädiktion stehen 35 verschiedene Modi zur Auswahl: 33 direktionale Modi, ein planarer Modus und einen Gleichanteilsmodus, engl. Direct Current (DC). Die direktionalen Modi basieren auf der linearen Extrapolation der Referenzwerte entlang einer von 33 Richtungen. Da in typischen Videosequenzen häufig horizontale und vertikale Strukturen vorkommen, wurde hierfür eine genauere Winkelauflösung als für andere Winkel gewählt [82, 140][18]. Wenn für eine zu verwendende Richtung die Referenzwerte für eine ganzzahlige Pixelposition eines zu prädizierenden Abtastwertes zwischen zwei ganzzahligen Pixelpositionen liegen, so erfolgt eine lineare Interpolation zwischen den zwei umgebenen Referenzwerten mit 1/32-Pel Genauigkeit. Die direktionalen Modi eignen sich gut für die Prädiktion von linearen Strukturen und Konturen [63].

Bei der Prädiktion mit dem planaren Modus erfolgt eine gewichtete Überlagerung von vier Referenzwerten ähnlich einer bilinearen Interpolation[19]. Dieser Modus eignet sich gut für Signale mit Helligkeitsverläufen sowie für Signale, die gleichzeitig horizontale und vertikale Strukturen enthalten.

Für mit dem DC-Modus prädizierte PB werden alle zu prädizierenden Abtastwerte auf den arithmetischen Mittelwert der Referenzwerte gesetzt. Es gibt eine Abweichung von diesem Prozess für direkt an bereits rekonstruierte Blöcke angrenzende Abtastwerte: Zur Vermeidung von starken Blockartefakten werden diese Abtastwerte als gewichtete Überlagerung des direkt angrenzenden Referenzwertes und des Mittelwerts der Referenzwerte prädiziert. Der Mittelwert wird hierbei stärker gewichtet. Durch dieses Vorgehen wird der Übergang an den Blockgrenzen geglättet. Der DC-Modus ist prädestiniert für einfarbige Signale. Gleichzeitig dient dieser Modus als Ausweichlösung für Blöcke, die mit den anderen Modi nur schlecht prädiziert werden können.

Damit die Intra-Prädiktion am Decoder durchgeführt werden kann, ist die Codierung von Seiteninformationen notwendig. Insbesondere wird

17 Es gibt drei verschiedene Varianten der Filterung [140]: Keine Filterung: Für Blöcke der Größe $4 \times 4$, DC-Modus, horizontale und vertikale Prädiktion. Normale Filterung: Für Blöcke größer als $4 \times 4$, für unterschiedliche Blockgrößen für unterschiedliche Richtungen. Starke Filterung: Für $32 \times 32$ Blöcke in Abhängigkeit der örtlichen Aktivität.

18 Implementierungsgründe sind eine weitere Ursache für die genauere Winkelauflösung.

19 Der Unterschied liegt in der Position und in der Gewichtung der verschiedenen Referenzwerte.

der verwendete Intra-Prädiktionsmodus übertragen. Hierzu wird zunächst eine Liste mit den drei wahrscheinlichsten Kandidaten, engl. Most Probable Modes (MPM), für den Intra-Prädiktionsmodus basierend auf den Modi von benachbarten Blöcken und verschiedener Standardwerte bestimmt [62]. Mit einem Bin wird signalisiert, ob der verwendete Intra-Modus zu den MPM gehört. Ist dieses der Fall, so wird ein Index für den richtigen Kandidaten aus der MPM-Liste mit Kontextmodellierung signalisiert. Wird stattdessen einer der 32 verbleibenden Modi verwendet, so wird hierfür ein Code mit fester Länge und ohne Kontextmodellierung[20] verwendet. Der Verzicht auf die Kontextmodellierung ist vorteilhaft für den Durchsatz bei der Entropiecodierung und bringt an dieser Stelle keinen wesentlichen Nachteil, da modellierbare Kontexte bereits durch die MPM-Liste erfasst sind.

Im Folgenden werden die von den allgemeinen Schilderungen der Hybridcodierung abweichenden Eigenschaften der Prädiktionsfehlercodierung in HEVC erläutert. In HEVC werden fünf verschiedene Transformationen in Ganzzahl-Arithmetik definiert. Hiervon nähern vier Transformationen für die vier Blockgrößen $4 \times 4$, $8 \times 8$, $16 \times 16$ und $32 \times 32$ die entsprechenden DCT-II-Transformationen mit Ganzzahlen an. Für die Blockgröße $4 \times 4$ wird zusätzlich eine Transformation definiert, welche die entsprechende DST annähert. Es konnte durch Han et al. in [37] gezeigt werden, dass die DST gut für eine einseitige Prädiktion aus einem Referenzabtastwert geeignet ist, weil hierbei der Fehler angrenzend an den bekannten Referenzbereich sehr klein ist und mit steigender Entfernung zum bekannten Bereich ansteigt. Dieses Beobachtung korrespondiert zu den Basisvektoren der DST. Folglich wird die $4 \times 4$-DST für die Prädiktionsfehler von intra-codierten Blöcken dieser Größe verwendet. Für größere Blöcke dominiert die bessere Dekorrelationseigenschaft der DCT. Deshalb wird für intra-codierte Blöcke der Größen $8 \times 8$ bis $32 \times 32$ die DCT verwendet.

Durch Narroschke wurde in [88] analysiert, dass die Prädiktionsfehlerwerte bereichsweise unterschiedlich starke statistische Abhängigkeiten voneinander haben und dass abhängig von der Stärke und der Ordnung der Abhängigkeiten die Codierung des Prädiktionsfehlers im Frequenzbereich oder im Ortsbereich effizienter ist. Diese Erkenntnis nutzend gibt es in HEVC den *Transform Skip*-Modus, mit dem adaptiv auf TU-Ebene die Transformation übersprungen werden kann [91]. Anschließend erfolgt eine Weiterverarbeitung des Prädiktionsfehlers im Ortsbereich. Der Transform Skip-Modus ist insbesondere für computergenerierte Videosignale nützlich [144].

20 CABAC Bypass-Modus

Der Prädiktionsfehler wird im Fall einer verlustbehafteten Codierung nach der Transformation quantisiert. Hierfür wird ein Quantisierer mit konstanter Stufenbreite verwendet. Die Stufenbreite wird über einen Quantisierungsparameter (QP) gesteuert. Für 8 Bit-Videosignale liegt dieser zwischen 0 und 51. Kleinere QP-Werte führen zu einer kleineren Stufenbreite, also einer feineren Quantisierung. Es gibt einen logarithmischen Zusammenhang zwischen dem QP und der Stufenbreite: Eine Erhöhung des QP um sechs führt jeweils zu einer Verdoppelung der Stufenbreite. In Abhängigkeit von der Bittiefe der Abtastwerte verändert sich der Wertebereich des QP. Für jedes zusätzliche Bit wird der Wertebereich um sechs Werte nach unten erweitert. Für 10 Bit-Videosignale kann der QP Werte im Bereich -12 bis 51 annehmen.

In der obigen Ausführung wird jeder Koeffizient des Prädiktionsfehlers gleichstark quantisiert. Aufgrund der Eigenschaften des menschlichen Wahrnehmungsapparats kann es jedoch sinnvoll sein, je nach Anwendung und zu codierendem Signal die höheren Frequenzanteile des Prädiktionsfehlers stärker zu quantisieren als die niedrigen Frequenzanteile. Optional kann hierfür in HEVC eine Gewichtungsmatrix für die Quantisierung verwendet werden. Am Encoder kann entweder eine von sechs vordefinierten Matrizen, jeweils eine für die Kombinationen aus Inter- und Intra-Codierung mit den drei Farbraumkomponenten, verwendet werden oder eine neue Matrix relativ zur vordefinierten Matrix signalisiert werden.

Bei HEVC kommen mehrere Verfahren für die Entropiecodierung zum Einsatz: Für Informationen auf der höchsten Ebene, zum Beispiel Network Abstraction Layer (NAL)-Header und Informationen, die die ganze Sequenz betreffen, werden Codes mit fester Codewortlänge verwendet. Hierdurch können diese Informationen leicht von Systemen verarbeitet werden, die nur diese Informationen auswerten, aber das Video nicht vollständig decodieren. Für *Parameter Sets* und die Konfiguration von Codierungsverfahren werden sowohl Codes mit fester Codewortlänge als auch solche mit variabler Codewortlänge, engl. Context-adaptive Variable Length Coding (CAVLC), verwendet. Diese Verfahren wurden gewählt, weil sie verglichen mit CABAC nicht so aufwendig zu decodieren sind. Dafür ist die Codierungseffizienz nicht so hoch wie bei CABAC. Das wird akzeptiert, da diese Informationen nur einen kleinen Anteil an der Gesamtdatenrate ausmachen. Ein Großteil der zu codierenden Informationen befindet sich auf Blockebene. Hierfür wird CABAC verwendet. Der Zustand von CABAC wird für jedes neue Slice beziehungsweise Tile zurückgesetzt. Wird die parallele Decodierbarkeit von CTU-Reihen aktiviert, engl. Wavefront Parallel Processing (WPP), so erfolgt eine Zurücksetzung

des Zustandes für jede neue CTU-Reihe [1]. Weitere Informationen zur Entropiecodierung in HEVC finden sich in [120].

*Encodersteuerung*

Dem HEVC-Standard entsprechende Bitströme können auf viele Weisen erzeugt werden. Die Codierung des gleichen Videos mit zwei unterschiedlichen Encodern, die beide gültige HEVC-Bitströme erzeugen, oder unterschiedliche Konfigurationen des gleichen Encoders können zu sich stark unterscheidenen aber dem Standard entsprechenden Bitströmen führen. Die Aufgabe der Encodersteuerung ist die Konfiguration eines Encoders, so dass unter den gegebenen Rahmenbedingungen ein HEVC-Bitstrom erzeugt wird. Hierbei umfasst die Encoderkonfiguration unter anderem die Partitionierung des Bildes, die Wahl der Codierungsverfahren sowie die weiteren einzustellenden Parameter des Encoders.

Es ist ersichtlich, dass die Rahmenbedingungen und damit auch die Encodersteuerung sehr anwendungsspezifisch sind. Zum einen hängt sie ab von den verfügbaren Ressourcen: Encoder, welche auf Hochleistungsservern oder in der Cloud ausgeführt werden, haben wesentlich mehr Rechenleistung zur Verfügung als Encoder auf mobilen Endgeräten wie beispielsweise Smartphones. Zum anderen werden durch die jeweilige Anwendung häufig zeitliche Randbedingungen gesetzt: Für die Offline-Speicherung von Videos, zum Beispiel für On-Demand-Video, sind bei der Encodierung auftretende Verzögerungen in der Regel vernachlässigbar. Selbst Encodierungszeiten von mehreren Wochen pro Spielfilm können akzeptabel sein. Bei Echtzeitübertragungen ist die zeitnahe Encodierung notwendig. Die genaueren Anforderungen hängen wiederum von der Anwendung ab. Während bei einer unidirektionalen Echtzeitübertragung, zum Beispiel für Fernsehübertragungen oder Streaming-Anwendungen, eine Verzögerung von einigen Sekunden unproblematisch sein kann, führt bei multidirektionalen Übertragungen, zum Beispiel für die Videotelefonie, bereits eine leichte Verzögerung zu einer Beeinträchtigung des Dienstes. Aus diesen Randbedingungen ergibt sich nicht nur, mit wie viel Rechenaufwand eine Encodierung durchgeführt werden kann, sondern auch, welche Signalanteile für die Prädiktion zur Verfügung stehen. In manchen Konfigurationen werden die Bilder einer Videosequenz so codiert, dass nur Bilder aus der Vergangenheit referenziert werden. In anderen Konfigurationen werden die Bilder innerhalb einer GOP zu einer hierarchischen Struktur umgeordnet, in der die Bilder aus einer Hierarchie-Ebene hintereinander codiert werden bevor die Bilder der nächsten Hierarchie-Ebene codiert werden. Hierdurch weicht die Codierungsreihenfolge von der Anzeigereihenfolge ab. Dadurch können

in der Zukunft angezeigte Bilder während der Codierung referenziert werden, wenn sie einer bereits codierten Hierarchie-Ebene zugeordnet sind[21].

Bei der Videocodierung werden in der Regel zwei gegenläufige Ziele verfolgt: Zum einen soll die Datenrate aus Effizienz- und Kostengründen möglichst klein sein. Zum anderen soll die Verzerrung ebenfalls möglichst gering, die Qualität also besonders hoch, sein. Typischerweise sind die Verzerrung und die Datenrate miteinander verknüpft: Eine Verringerung der Verzerrung führt oft zu einer Erhöhung der Datenrate. Das gleichzeitige Optimieren der Verzerrung und der Datenrate ist deshalb notwendig. Hierfür wird eine Raten-Verzerrungs-Optimierung, engl. Rate-Distortion Optimization (RDO), eingesetzt [115]. Bei der RDO werden die Datenrate $R$ und die Verzerrung, engl. *Distortion*, $D$ mit dem Lagrange-Faktor $\lambda$ verknüft, um die Rate-Distortion-Kosten $C_{RD}$ zu berechnen:

$$C_{RD} = D + \lambda R \tag{2.2}$$

Bei gegebener Rate und Verzerrung ergibt sich für die Rate-Distortion-Kosten eine Arbeitskurve in Abhängigkeit von $\lambda$. Dieser Parameter ist anwendungsspezifisch, da die Wichtigkeit einer geringeren Verzerrung und einer geringeren Datenrate je nach Anwendung variieren kann. Die RDO kann durch die Notwendigkeit, sehr viele Kombinationen aus Partitionierung, Wahl der Codierungsverfahren, Einstellung der Parameter auszuprobieren, sehr rechenaufwendig sein [66]. Da je nach verfügbaren Ressourcen nur ein begrenzter Aufwand für die Codierung eines Videos im gegebenen Zeit- und Finanzbudget möglich ist, haben verschiedene Anwendungen unterschiedliche Arbeitskurven für die Rate-Distortion-Kosten.

## 2.2 MASCHINELLES LERNEN

> Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed. [104]
>
> Arthur Samuel (IBM)
> 1959

In diesem Abschnitt werden nach einigen grundlegenden Einführungen zum Thema maschinelles Lernen insbesondere die Konfiguration neuronaler Netzwerke sowie das Training neuronaler Netzwerke vertieft.

21 Vgl. die Konfigurationen *Low Delay* und *Random Access* in den im Kontext der Standardisierung verwendeten Testbedingungen, engl. Common Test Conditions (CTC) [54].

Vierzig Jahre nach der im Epigraph wiedergegebenen noch weiten Definition von Samuel definiert Mitchell ein gut formuliertes maschinelles Lern-Problem wie folgt: „Ein Computerprogramm wird als lernend von der Erfahrung $E$ bezüglich einer Aufgabe $T$ und einer Leistungsmetrik $P$ bezeichnet, wenn dessen Leistung für $T$, gemessen mittels $P$, sich mit der Erfahrung $E$ verbessert.“[22] Diese Definition kann auf die vorliegende Arbeit angewendet werden. Hier ist die Aufgabe $T$ die Abtastwertprädiktion im Rahmen der vorgeschlagenen Verfahren, die Leistungsmetrik $P$ zum Beispiel die Verzerrung, die Datenrate oder die Rate-Distortion (RD)-Kosten sowie die Erfahrung $E$ die Menge an Beispielen, welche für das Lernen verwendet werden. Die Menge an Beispielen, die während des Lernvorgangs verwendet wird, wird oft als Datensatz bezeichnet. Einzelne Beispiele aus dem Datensatz werden auch als Datenpunkte bezeichnet. In modernen Definitionen [31] umfasst ein maschineller Lernalgorithmus

- ein Modell, welches gelernt wird,
- einen Optimierungsalgorithmus, der zum Lernen eingesetzt wird,
- eine Kostenfunktion, mit welcher der Lernerfolg bewertet wird
- und einen Datensatz, der während des Lernens verarbeitet wird.

Maschinelle Lernprogramme verarbeiten in der Regel Beispiele, welche jeweils aus einer Menge an Merkmalen, engl. *Features*, zusammengesetzt sind und erzeugen basierend auf den verarbeiteten Beispielen ein Ausgangssignal[23]. Während des Lernvorgangs wird der Algorithmus fortlaufend angepasst, er „lernt“, um gemessen mit der gewählten Leistungsmetrik die gestellte Aufgabe besser zu lösen.

Die Merkmale werden aus den zu verarbeitenden Daten gewonnen und können diese Daten unterschiedlich stark abstrahieren: Bezogen auf Bildsignale könnten wenig abstrahierende Merkmale die Abtastwerte des betrachteten Bildsignals sein. Mit zunehmendem Grad an Abstrahierung könnten aus den Abtastwerten geometrische Strukturen und Objekte, die den Bildinhalt beschreiben, werden.

In Abhängigkeit davon, was für Erfahrungen während des Lernvorgangs verwendet werden, können maschinelle Lernverfahren in über-

22 „Well posed Learning Problem: A computer program is said to learn from experience $E$ with respect to some task $T$ and some performance measure $P$, if its performance on $T$, as measured by $P$, improves with experience $E$.“– Tom Mitchell, 1997 [85].

23 Nicht betrachtet werden im Rahmen dieser Arbeit Ansätze, in denen es zusätzlich zu der Verarbeitung des Datensatzes eine Rückkopplung zwischen dem Lernalgorithmus und dem Datensatz gibt. Zu dieser Art von Verfahren gehören Reinforcement Learning-Ansätze. Hierbei verursachen durch den Lernalgorithmus ermittelte Ausgangswerte eine Veränderung der nachfolgend als Eingang verarbeiteten Beispiele.

wachte Lernverfahren und unüberwachte Lernverfahren[24] unterschieden werden. Bei unüberwachten Lernverfahren werden während des Lernvorgangs die Merkmale aus dem Datensatz genutzt, um relevante Eigenschaften der dem Datensatz zu Grunde liegenden Quelle zu bestimmen. Typische Anwendungen umfassen Hauptkomponentenanalyse, Clusteranalyse, Entrauschung oder Synthese. Bei überwachten Lernverfahren umfassen die verwendeten Erfahrungen zusätzlich zu den als Eingang verwendeten Merkmalen ein oder mehrere gewünschte Ergebnisse der Lernvorgangs, welche mit den Merkmalen assoziiert sind. Das gewünschte Ergebnis wird teilweise auch als Label, Ziel oder Ground Truth bezeichnet. Somit kann als Ziel dieser Gruppe an Verfahren das Lernen einer Abbildung von Eingangs- auf Ausgangsdaten interpretiert werden. Typische Anwendungen umfassen Klassifikation und Regression. Die Grenze zwischen überwachten und unüberwachten Lernalgorithmen ist weder klar definiert noch in allen Fällen eindeutig. Goodfellow et al. argumentieren, dass sich umfangreiche unüberwachte Lernverfahren in eine Kombination aus einer endlichen Anzahl an weniger umfangreichen überwachten Lernalgorithmen überführen lassen [31].

In Abhängigkeit der durch das Modell gelernten Zusammenhänge können generative und diskriminative Modelle unterschieden werden [90]: Während bei generativen Modellen die Verbundwahrscheinlichkeit der Ein- und Ausgangssignale gelernt werden, werden bei diskriminativen Modellen die bedingten Wahrscheinlichkeiten für das Ausgangssignal gegeben das Eingangssignal gelernt. Obgleich aus der Verbundwahrscheinlichkeit mittels der Bayes-Regel die bedingte Wahrscheinlichkeit ermittelt werden kann – und die generativen Modelle somit umfassender als die diskriminativen Modelle erscheinen – sind in der Praxis für die Aufgabe der Abbildung eines Eingangssignals auf ein Ausgangssignal die diskriminativen Modelle effizienter [90].

Das Ziel von maschinellen Lernalgorithmen ist in der Regel nicht nur, die jeweilige Aufgabe auf dem für den Lernvorgang verwendeten Datensatz, dem sogenannten Trainingsdatensatz, gut zu lösen, sondern die gleiche Aufgabe auch auf neuen, im Trainingsdatensatz nicht enthaltenen, Beispielen gut zu erfüllen. Diese neuen Daten werden auch als Testdaten bezeichnet. Diese Fähigkeit wird bei maschinellen Lernansätzen als Generalisierung bezeichnet[25]. Während des Lernvorgangs kann der Algorithmus nur auf den Trainingsdaten evaluiert werden.

24 Die Begriffe beruhen auf der Analogie eines Schülers, der während des Lernens von einem Lehrer überwacht beziehungsweise angeleitet wird oder nicht.

25 Hier unterscheiden sich maschinelle Lernverfahren von Optimierungsverfahren. Bei letzteren werden lediglich die bereits vorliegenden Daten betrachtet. Beim Lernen wird angenommen, dass die Testdaten während des Trainings nicht zur Verfügung stehen, so dass keine Optimierung auf diese möglich ist.

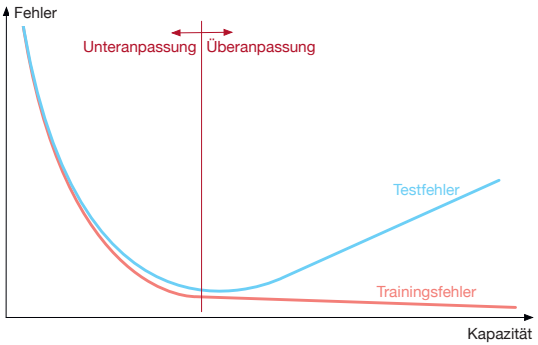



Abbildung 2.5: Zusammenhang zwischen Kapazität und Über- und Unteranpassung. In Anlehnung an [31].

Deshalb wird der Algorithmus zunächst während des Lernvorgangs so angepasst, dass der Fehler (vgl. Abschnitt 2.2) auf den Trainingsdaten (Trainingsfehler), $e_{\text{train}}$, minimiert wird. Eine Generalisierung ist möglich, wenn Trainings- und Testdaten (näherungsweise) die gleichen statistischen Eigenschaften haben, wenn die Beispiele in den Datensätzen jeweils unabhängig voneinander sind und wenn die Datensätze hinreichend umfangreich sind um die statistischen Eigenschaften des datenerzeugenden Prozesses abzubilden [31]. Eine gute Generalisierung liegt vor, wenn der Abstand zwischen $e_{\text{Training}}$ und dem Fehler auf den Testdaten (Testfehler), $e_{\text{Test}}$, möglichst klein wird. Die Differenz wird als Generalisierungsfehler, $e_{\text{Gen}}$[26], bezeichnet und geht idealerweise gegen Null:

$$e_{\text{Gen}} = e_{\text{Training}} - e_{\text{Test}} \to 0. \tag{2.3}$$

Mit Bezug auf die genannten Fehler können zwei Probleme während der Entwicklung von maschinellen Lernalgorithmen auftreten: Unteranpassung und Überanpassung. Eine Unteranpassung liegt vor, wenn der Trainingsfehler während des Lernvorgangs nicht hinreichend klein wird. Eine Überanpassung liegt vor, wenn der Trainingsfehler zwar hinreichend klein wird, aber hierbei der Generalisierungsfehler zu groß wird. Der Arbeitspunkt von maschinellen Lernalgorithmen mit Bezug auf Unter- und Überanpassung lässt sich unter anderem über die Kapazität des Modells, auf den der Lernalgorithmus angewendet wird, kontrollieren.

Bei der Kapazität handelt es sich um einen (nicht präzise definierten) Begriff, der die Fähigkeit des Modells, komplexe und vielfältige Auf-

26 In der Literatur wird teilweise auch der Testfehler abweichend von der hier verwendeten Nomenklatur als Generalisierungsfehler bezeichnet. Die Motivation hiervon ist, dass dieser Fehler bei der Generalisierung auf ungesehene Daten auftritt.

gaben zu lernen, beschreibt [31]. Ist die Kapazität zu klein, so können die Eigenschaften des den Daten zugrunde liegenden Prozesses nicht hinreichend genau erfasst werden und es kommt zu Unteranpassung. Ist die Kapazität hingegen zu groß, so werden nicht die Eigenschaften des Prozesses gelernt. Stattdessen werden die Eigenschaften von den konkreten Beispielen des Trainingsdatensatzes zu detailliert (d.h. auswendig) gelernt. In Folge dessen ist keine Generalisierung auf ungesehene Daten möglich. Somit kommt es zu einer Überanpassung. Dieser Zusammenhang wird in Abbildung 2.5 grafisch veranschaulicht. Die Kapazität kann auf unterschiedliche Weisen angepasst werden, zum Beispiel durch die Wahl des Modells oder durch Parameter dieses Modells. Ein als *Ockhams Rasiermesser*, engl. *Occam's Razor*, benannter, auf maschinelles Lernen anwendbarer, Grundsatz besagt, dass von mehreren gleich guten Lösungen (hier: Modellen mit den jeweiligen Konfigurationen) die einfachste (hier: die mit der geringsten Kapazität) zu bevorzugen ist [6, 31, 106]. Für eine formellere Betrachtung der Kapazität sei der Leser auf [31] verwiesen.

Neben der Kapazitätsanpassung werden oft Regularisierungsansätze zur Verbesserung der Generalisierung eingesetzt. *Regularisierung* wird als Einflussnahme, zum Beispiel über zusätzliche Terme in der Kostenfunktion, auf den Lernalgorithmus mit dem Ziel, den Generalisierungs- aber nicht den Trainingsfehler zu reduzieren, definiert.

Maschinelle Lernalgorithmen haben in der Regel sogenannte *Hyperparameter*, welche nicht durch den Lernalgorithmus selbst – oder präziser: das beinhaltete Optimierungsverfahren – bestimmt werden. Stattdessen sind diese Parameter durch den Entwickler des Lernalgorithmus oder durch einen umhüllenden zweiten Algorithmus zu bestimmen. Mit diesen Hyperparametern können zum Beispiel die Kapazität des Modells oder die Funktionsweise des Optimierungsverfahrens beeinflusst werden. Von den für die Hyperparameter gesetzten Werten kann die Leistung des Lernalgorithmus grundlegend abhängen; sie können den Ausschlag zwischen einer sehr guten Leistung und einem Nicht-Funktionieren geben. Eine Optimierung der Hyperparameter auf den Trainingsdatensatz ist nicht sinnvoll, da dieses nur zu einer Überanpassung führen würde. So ließe sich in vielen Fällen zwar durch eine Erhöhung der Kapazität der Trainingsfehler reduzieren aber gleichzeitig würde der Generalisierungsfehler steigen. Eine Optimierung auf den Testdatensatz ist ebenfalls nicht sinnvoll. Zum einen ist dieses nicht sinnvoll, da konzeptionell angenommen wird, dass die Testdaten während der Entwicklung des Lernalgorithmus, hierzu gehört der Lernvorgang, unbekannt sind. Zum anderen ist dieses ebenfalls nicht sinnvoll, da dieses auch zu einer Überanpassung, und zwar auf den Testdatensatz, führen würde. Folglich wird für die Optimierung der Hyperparameter oft ein dritter

Datensatz, der als *Validierungsdatensatz* bezeichnet wird, verwendet. Typischerweise werden die für die Entwicklung des Lernalgorithmus zur Verfügung stehenden Daten zufällig in einen Trainings- und in einen Validierungsdatensatz aufgeteilt, oft in einem Verhältnis von 80 zu 20[27]. Aufgrund des großen Einflusses von Hyperparametern auf die Leistung maschineller Lernalgorithmen und dem oft immensen Aufwand für eine gute Hyperparameteroptimierung ist für eine sinnvolle Vergleichbarkeit von verschiedenen Lernalgorithmen sicherzustellen, dass die jeweiligen Hyperparameter vergleichbar gut optimiert wurden.

Einen weiteren fundamentalen Grundsatz im Kontext des maschinellen Lernens stellt das *No-Free-Lunch-Theorem* [141] (sinngemäß: Nichts-ist-umsonst-Theorem) dar. Es besagt, dass alle Lernalgorithmen die gleiche Leistung erreichen, wenn über alle denkbaren Aufgaben für Lernalgorithmen gemittelt wird. Folglich gebe es keine guten oder schlechten Lernalgorithmen. Erst durch die Spezialisierung eines Lernalgorithmus auf eine bestimmte Aufgabe und durch das Treffen geeigneter Annahmen, zum Beispiel in Form von Hyperparametern, über die zu verarbeitenden Daten sei maschinelles Lernen sinvoll [31].

Die Anwendungsmöglichkeiten für maschinelle Lernalgorithmen sind vielfältig. Im Folgenden werden einige Beispielanwendungen mit Schwerpunkt auf die in dieser Arbeit betrachteten neuronalen Netzwerke genannt.

*Klassifikation:* Eine der ersten Anwendungen, welche seit der Arbeit [61] von Krizhevsky von auf neuronalen Netzwerken basierenden Ansätzen dominiert wird, ist die Klassifikation. Alle Gewinner-Verfahren in den letzten Jahren des Wettbewerbs *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) [103], bei dem Objekte in Bildern erkannt werden müssen, basieren auf neuronalen Netzwerken. Die an Klassifikationsalgorithmen gerichtete Aufgabe ist die Zuordnung eines Beispiels in eine Kategorie aus einer Menge an möglichen Kategorien. Teilweise erfolgt auch eine Ausgabe der Wahrscheinlichkeiten für die Kategorien. Beispiele sind die Objekterkennung im oben genannten Wettbewerb oder die Gesichtserkennung [99]. Auch im Bereich der Videocodierung werden verstärkt Aufgaben der Encodersteuerung als Klassifikationsproblem formuliert [66].

*Regression:* Bei Regressionsproblemen werden basierend auf dem Eingangssignal einer oder mehrere numerische Werte als Ausgang bestimmt. Auch hier werden oft maschinelle Lernansätze zur Lösung dieser Probleme angewendet [64]. Beispiele umfassen die Altersbestimmung aus Bildern [134, 147] sowie die Poseschätzung [128].

27 Die Daumenregel des 80-zu-20-Verhältnisses beruht auf dem *Paretoprinzip* [98], in welchem proklamiert wird, dass 80% des Ergebnisses mit 20% des Aufwands erreichbar sind.

*Interpretation:* Bei diesem Anwendungsfall ist das Ziel die Erfassung des Inhalts von zu verarbeitenden Daten. Konkrete Beispiele sind die Spracherkennung, hier ist die Eingabe die Audioaufnahme von Sprache und die gewünschte Ausgabe die gesprochenen Wörter als Text [40], oder die Extraktion von Adressinformationen aus Straßenaufnahmen [32]. Ebenfalls werden Lernalgorithmen zur Interpretation einer ganzen Szene basierend auf einer Bild- oder Videoaufnahme [43] verwendet.

*Übersetzung:* Ein weiterer Anwendungsfall ist die Übersetzung von Text zwischen verschiedenen Sprachen [4]. Teilweise geschieht dieses auch in Kombination mit Algorithmen zur Spracherkennung in Audiosignalen oder Texterkennung in Bildern.

*Anomaliedetektion:* Das Ziel der Anomaliedetektion ist die Detektion von Abweichungen von einem erwarteten, als normal angesehenen, Verhalten in den zu verarbeitenden Daten. Beispiele finden sich im Bereich der Sicherheitstechnik für die Auswertung von Überwachungsvideos oder bei Sicherheitsmechanismen zur Verhinderung von Kreditkartenbetrug.

*Synthese/Generierung:* Die Synthese von Signalen unter gegebenen Randbedingungen und optional Eingangssignalen ist ein weiteres Anwendungsgebiet für Lernalgorithmen. Typische Beispiele umfassen die Sprachsynthese gegeben eines zu sprechenden Textes [21, 148] und die Bewegungssynthese für Animationen, beispielsweise für die Filmproduktion oder Computerspiele [41]. Durch die automatisierte Animation lässt sich der immense Aufwand für eine manuelle Animation umgehen. Die generative Bildmodellierung und Inpainting-Verfahren wurden bereits in der Einleitung dieser Arbeit thematisiert [27, 33, 36, 124]. Ebenfalls gibt es Arbeiten zur Stilübertragung [53]. Hierbei wird zum Beispiel der Stil eines Gemäldes auf ein Foto übertragen.

*Entrauschung:* Die Entrauschung von rauschbehafteten Signalen, zum Beispiel Bildern [132, 143, 149] oder Datenreihen, ist ebenfalls eine Möglichkeit für den Einsatz von Lernalgorithmen. Die Ursachen von Rauschen können hierbei sehr vielfältig und damit schwierig zu erfassen sein.

*Superresolution:* Bei Superresolution-Ansätzen wird die Erzeugung eines Bildes mit hoher örtlicher Auflösung aus einem niedrig aufgelösten Bild [23, 55, 72] umgesetzt. Die Schwierigkeit ist hierbei die Generierung oder Wiederherstellung von feinen Details und Texturen, welche in dem Bild mit niedriger Auflösung nicht oder nur teilweise vorhanden sind.

### *Neuronale Netzwerke*

Im Rahmen dieser Arbeit und heutzutage typischerweise auch für die zuvor genannten Anwendungen werden (künstliche) neuronale Netz-

werke[28] als Modell für die Lernalgorithmen verwendet. Deshalb werden neuronale Netzwerke in diesem Abschnitt eingeführt.

Obgleich neuronale Netzwerke erst seit ungefähr 2012 [61] enorme Aufmerksamkeit außerhalb der wissenschaftlichen Gemeinschaft erhalten haben, sind sie seit Jahrzehnten bekannt. Zunächst wird ein kompakter Überblick über neuronale Netzwerke im Wandel der Zeit gegeben. Bereits vorkommende relevante Begriffe, die noch nicht eingeführt wurden, werden im Anschluss eingeführt. Die Ausführungen folgen der historischen Abhandlung [106] von Schmidhuber.

Die ersten neuronalen Netzwerke waren Implementierungen einer linearen Regression, welche seit Jahrhunderten bekannt ist [29]. Die Idee der neuronalen Netzwerke wurde erstmals in den 40er-Jahren ohne Lernen [81] und für unüberwachtes Lernen [39] vorgeschlagen. Im Kontext des überwachten Lernens wurden neuronale Netzwerke in den darauffolgenden Jahrzehnten diskutiert [102, 138]. Nicht-lineare Aktivierungsfunktionen werden seit 1965 [52] für neuronale Netzwerke verwendet. Das Konzept von tiefen neuronalen Netzwerken ist seit den 70er Jahren bekannt. 1971 schlug Ivanhnenko ein Netzwerk mit acht Schichten vor [51]. Fukushima schlug in den Jahren 1979 und 1980 ein tiefes Faltungsnetzwerk vor, bei dem die Anzahl an zu lernenden Parametern durch die Verwendung geteilter Werte, engl. *weight sharing*, deutlich gegenüber vollverbundenen neuronalen Netzwerken reduziert wurde. Vorstufen des *Backpropagation*-Algorithmus wurden 1970 publiziert, ein erster Einsatz für neuronale Netzwerke fand 1981 statt [137].

Neuronale Netzwerke bestehen, wie auch die als Inspiration dienenden Gehirne, aus vielen miteinander verbundenen Neuronen. In Abbildung 2.6 ist ein (künstliches) Neuron illustriert. Basierend auf der Summe von mehreren mit $w_i$ gewichteten Eingangssignalen $i_i$ und einem Gleichanteil $b$, engl. *bias*, welche in dem Neuron mit einer (typischerweise nichtlinearen) Aktivierungsfunktion $a$ verarbeitet werden, gibt es ein als Aktivierung bezeichnetes Signal $o_i$ am Ausgang. Die Übertragungsfunktion des Neurons, also $o(i)$[29], lässt sich wie folgt formulieren:

$$o(i) = a(w^T i + b) \tag{2.4}$$

Das Ausgangssignal kann dann wiederum als eines der Eingangssignale für ein anderes Neuron verwendet werden. Die Gewichte sind das wesentliche, für die Funktionalität eines Netzwerkes verantwortliche,

28 Der Name zeigt an, dass die Netzwerke von der Funktionsweise des Gehirns, oder einiger Teilaspekte desselben, inspiriert wurden. Dies bedeutet jedoch nicht, dass mit neuronalen Netzwerken ein Gehirn „nachgebaut"werden soll.

29 $o(i)$ hängt zwar auch von den veränderlichen Größen $w$ und $b$ ab. Diese werden jedoch nur während des Trainings des Netzwerkes verändert und werden zum Zeitpunkt der eigentlichen Anwendung des Netzwerkes als unveränderlich betrachtet.

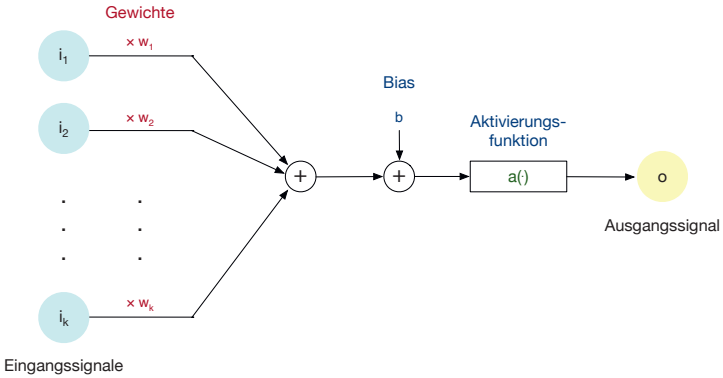



Abbildung 2.6: Schematische Darstellung eines (künstlichen) Neurons.

Merkmal des Neurons. Während des Trainings werden die Gewichte der Neuronen sukzessive angepasst, um die Leistung des Netzwerkes für die gestellte Aufgabe zu verbessern.

Aus mehreren in sogenannten Schichten angeordneten Neuronen werden neuronale Netzwerke aufgebaut. Eine Darstellung des grundlegenden Aufbaus neuronaler Netzwerke findet sich in Abbildung 2.7. Die erste Schicht des Netzwerkes wird als Eingangsschicht bezeichnet, die letzte als Ausgangsschicht. Zwischen der Eingangs- und der Ausgangsschicht befinden sich eine oder mehrere verborgene Schichten, engl. *hidden layer*. Als tiefe neuronale Netzwerke, teilweise auch mittels *Deep Learning* referenziert, werden Netzwerke mit mehr als einer verborgenen Schicht bezeichnet [121]. Moderne neuronale Netzwerke haben zwischen fünf und über 100 verborgene Schichten [13]. Die Neuronen einer Schicht sind mit einigen oder allen Neuronen der vorherigen Schicht über die Gewichte $w$ verbunden.

Gäbe es in Gleichung 2.4 nicht die Aktivierungsfunktion $a$, dann entspräche die Gleichung lediglich der gewichteten Überlagerung der Eingangssignale mit einer zusätzlichen Addition des Bias. Der Wertebereich am Ausgang entspräche den kompletten reellen Zahlen. Inspiriert durch die Funktionsweise von Neuronen im Gehirn, welche nur dann ein Ausgangssignal erzeugen, wenn die Überlagerung der Eingangssignale über einem Grenzwert liegt, umfassen auch künstliche Neuronen eine Aktivierungsfunktion.

Die Anforderungen umfassen mindestens die folgenden vier Aspekte: Die Aktivierungsfunktion sollte nichtlinear sein. Die Nichtlinearität wird für das Lernen komplexer Zusammenhänge benötigt. Würden lediglich

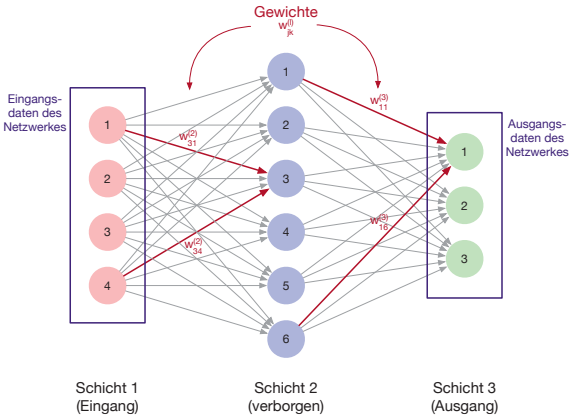



Abbildung 2.7: Schematische Darstellung eines (künstlichen) neuronalen Netzwerkes. In Anlehnung an [121].

lineare Aktivierungsfunktionen verwendet, so könnten selbst bei mehreren hintereinander geschalteten Neuronen nur lineare Zusammenhänge durch das Netzwerk erlernt werden. Sukzessive Änderungen an den Eingangssignalen sollten zu einer sukzessiven Änderung der Aktivierung führen. Durch diese Konsistenz sollen kleine Änderungen des Eingangssignals, zum Beispiel Rauschen, des Netzwerkes nicht zu einem sich stark ändernden Ausgangssignal, beispielsweise einer anderen Klasse bei Klassifizierungsaufgaben, führen. Die Ausgangswerte der Aktivierungsfunktion sollten nicht zu groß werden. Die Aktivierungsfunktion sollte differenzierbar sein und die Gradienten sollten sinnvoll für den nachfolgend beschriebenen Optimierungsalgorithmus verwendbar sein. Hierfür sollten die Gradienten weder zu groß noch zu klein werden.

Im Folgenden werden vier typische Aktivierungsfunktionen $a(x)$ für Eingangssignale $x$ diskutiert. Diese Funktionen werden zusätzlich in Abbildung 2.8 dargestellt.

Die Sigmoid-Funktion ist definiert als:

$$a_{\text{Sigmoid}}(x) = \frac{1}{1+e^{-x}} \tag{2.5}$$

Aufgrund des Wertebereichs des Ausgangssignal zwischen Null und Eins wird diese Aktivierungsfunktion gerne für Anwendungen verwendet, in dem am Ausgang eine Wahrscheinlichkeit, die ebenfalls zwischen Null und Eins liegt, gefordert ist. Eine weitere Anwendungsmöglichkeit stellt

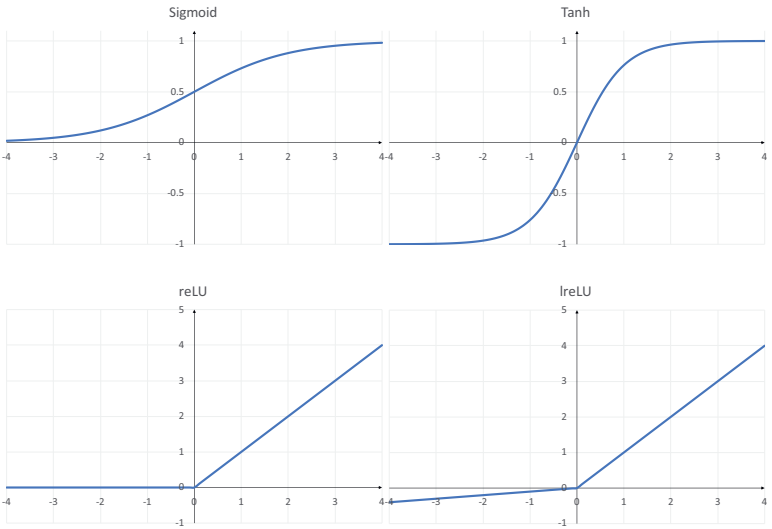



Abbildung 2.8: Übersicht über die vier Aktivierungsfunktionen Sigmoid, tanh, ReLU und lreLU

die binäre Klassifikationen dar[30]. Die Sigmoid-Aktivierungsfunktion erfüllt viele der an Aktivierungsfunktionen gestellten Anforderungen: Sie ist nichtlinear, es kommt zu sukzessiven Änderungen des Ausgangssignals bei sukzessiven Änderungen des Eingangssignals, das Ausgangssignal ist auf den Wertebereich Null bis Eins begrenzt und die Funktion ist differenzierbar. Die Ableitung der Funktion ist für Eingangswerte mit kleinem Betrag groß. Das ist nützlich für den später eingeführten Backpropagation-Algorithmus. Das Hauptproblem dieser Aktivierungsfunktion ist, dass die Ableitung für Eingangssignale mit großem Betrag sehr klein ist. Das kann während der Optimierung des Modells zu einem Problem werden, wenn das Modell sich in einem Zustand befindet, in dem Neuronen solche großen Werte ausgeben. Dann kann es passieren, dass diese Werte wegen der verschwindenden Gradienten sich während der Optimierung nicht verändern und der Optimierungsalgorithmus zu keiner Verbesserung des Modells mehr führt. Dieses lässt sich als Steckenbleiben während des Trainings vorstellen[31]. Aufgrund des be-

30 Eine Verallgemeinerung für nicht-binäre Klassifikationsaufgaben bildet die Softmax-Aktivierungsfunktion

31 Der Nachteil der Sättigung bei Werten mit großem Betrag kann vernachlässigt werden, wenn die Sigmoid-Funktion als Aktivierungsfunktion in der letzten Schicht des Netzwerkes verwendet wird und mit einer geeigneten, zum Beispiel logarithmischen, Kostenfunktion kombiniert wird.

schriebenen Nachteils wird die Sigmoid-Aktivierungsfunktion in der Regel nicht mehr für verborgene Schichten in Feed-Forward-Netzwerken verwendet. Für andere, in dieser Arbeit nicht betrachtete, Netzwerkarten wird diese Aktivierungsfunktion dennoch weiter verwendet. Der Grund hierfür ist, dass andere Aktivierungsfunktionen wie die nachfolgend betrachtete und für Feed-Forward-Netzwerke präferierte ReLU-Funktion hier aus anderen Gründen nicht eingesetzt werden können.

Der Tangens hyperbolicus (*tanh*) ist eine verschobene und skalierte Version der Sigmoid-Aktivierungsfunktion:

$$a_{\tanh}(x) = 2a_{\text{Sigmoid}}(2x) - 1 \tag{2.6}$$

$$= \frac{2}{1 + e^{-2x}} - 1 \tag{2.7}$$

Die wesentlichen Änderungen gegenüber der Sigmoid-Aktivierungsfunktion sind, dass zum einen die Steigung größer ist und zum anderen der Sättigungsbereich für betragsgroße negative Eingangssignale nicht bei Null liegt. Die erste Eigenschaft kann sinnvoll sein, wenn für die Optimierung größere Gradienten benötigt werden. Die zweite Eigenschaft bewirkt, dass das Ausgangssignal – und damit das Eingangssignal für nachfolgende Neuronen – für den relevanten Sättigungsbereich nicht bei Null liegt, wodurch die Gewichte der nachfolgenden Neuronen bedeutungslos würden.

Am häufigsten in modernen Netzwerken werden die gleichrichtende Aktivierungsfunktion, engl. *Rectifier Linear Unit* (ReLU), oder Varianten ebenjener verwendet. Sie ist definiert als:

$$a_{\text{reLU}}(x) = \max(0, x) \tag{2.8}$$

$$= \begin{cases} x & \text{wenn } x > 0 \\ 0 & \text{sonst} \end{cases} \tag{2.9}$$

Obgleich die ReLU stückweise linear ist, ist sie als ganzes nichtlinear. Abgesehen vom Ursprung ist die Funktion differenzierbar. Durch die Nichtdifferenzierbarkeit im Ursprung verletzt diese Aktivierungsfunktion die gestellten Anforderungen. Dieses kann jedoch vernachlässigt werden, da es sich beim Ursprung über keinen typischerweise während des Trainings auftretenden Arbeitspunkt handelt [31]. Für positive Werte des Eingangssignals wird das Signal ohne Veränderung ausgegeben, somit erfüllt die Funktion die diesbezüglichen Anforderungen. Für negative Werte ist der Ausgabewert der Funktion immer Null. Dieses kann zu „sterbenden“Neuronen führen, die aufgrund des fehlenden Gradienten diesen Zustand während des Trainings nie verlassen können und immer

Nullen ausgeben. Hierdurch können in ungünstigen Fällen größere Teile eines Netzwerkes dauerhaft deaktiviert werden. Dennoch ist die Funktion als Aktivierungsfunktion beliebt, da sie sehr einfach zu berechnen ist – es ist lediglich die Überprüfung eines Vorzeichens notwendig – und zu guten Lernergebnissen führt.

Die durchlässige gleichrichtende Aktivierungsfunktion, engl. *Leaky Rectifier Linear Unit* (lreLU), löst das Problem der toten Neuronen, welches bei der normalen ReLU-Aktivierungsfunktion besteht, in dem negativen Eingangswerten ein sehr kleiner, nicht konstanter Wert zugewiesen wird:

$$a_{\text{lreLU}}(x) = \begin{cases} x & \text{wenn } x > 0 \\ cx & \text{sonst} \end{cases} \tag{2.10}$$

mit einer Konstanten $c$, die typischerweise positiv und hinreichend klein, zum Beispiel $c = 0.01$, ist. Hier wird während der Optimierung der Netzwerkgewichte die Möglichkeit erhalten, die verbleibenden kleinen Gradienten zur Verschiebung des Arbeitspunkts aus dem toten Bereich zu nutzen.

Die Art der Neuronen in der letzten Schicht wird häufig in Abhängigkeit der durch das Netzwerk zu lösenden Aufgabe gewählt. Lineare Neuronen, hiermit werden Neuronen ohne Aktivierungsfunktion, also einer Übertragungsfunktion $o(i) = w^T i + b$, bezeichnet werden oft für Aufgaben verwendet, bei denen am Netzwerkausgang ein quadratischer Fehler minimiert werden soll. Neuronen mit Sigmoid-Aktivierungsfunktionen werden für binäre Klassifikationen oder andere Aufgaben, bei denen am Ausgang ein Wahrscheinlichkeitswert erwünscht ist, verwendet. Als Erweiterung werden Neuronen mit Softmax-Aktivierungsfunktion für Klassifikationen mit mehr als zwei Klassen verwendet. Für den Ausgang $j$ mit $j \in [1, J]$ erfolgt die Berechnung zu:

$$a_{\text{Softmax}}(x_j) = \frac{e^{x_j}}{\sum_{i=1}^{J} e^{x_i}} \tag{2.11}$$

Durch die Berechnung ist sichergestellt, dass alle Ausgangswerte zwischen Null und Eins liegen und die Summe über alle Ausgangswerte Eins ergibt. Hierdurch sind die Ausgangswerte als Wahrscheinlichkeiten sinnvoll interpretierbar.

In Abhängigkeit der verwendeten Neuronen und ihrer Verbindung zur vorherigen Schicht werden unterschiedliche Arten von Schichten unterschieden. Die im Rahmen dieser Arbeit verwendeten voll-verbundenen Schichten und Faltungsschichten werden im Folgenden erläutert. Für weitere Arten von Schichten sei der Leser auf die Übersichten von Goodfellow et al. in [31] und von van Veen in [131] verwiesen.

In voll-verbundenen Schichten sind alle Neuronen aus einer Schicht mit allen Neuronen der vorherigen Schicht verbunden. Der Ausgang $\boldsymbol{o_l}$ einer Schicht $l$ lässt sich als Erweiterung von Gleichung 2.4 zu

$$\boldsymbol{o_l} = a(W^T \boldsymbol{o_{l-1}} + \boldsymbol{b}) \tag{2.12}$$

mit einer elementweise angewandten Aktivierungsfunktion $a$, einer Gewichtsmatrix $W$ und einem Bias-Vektor $\boldsymbol{b}$ formulieren.

Bei Faltungsschichten gibt es anstatt einer großen Gewichtsmatrix $\boldsymbol{W}$ wie in Gleichung 2.12 eine Anzahl an Faltungsmatrizen, mit denen das Eingangssignal gefaltet wird. Faltungsschichten werden häufig für zweidimensionale Eingangsdaten des neuronalen Netzwerkes, zum Beispiel Bilder, verwendet. Die Faltungsmatrizen haben oft die gleiche Dimensionalität wie die Eingangsdaten für die Faltungsschicht: Bestehen die Eingangsdaten beispielsweise aus zweidimensionalen Bildern mit drei Kanälen, so sind die Faltungsmatrizen ebenfalls dreidimensional mit einer Tiefe von drei. Der Ausgang einer Faltungsschicht mit der Faltungsmatrix $W_F$ lässt sich mit $*$ als Faltungsoperator als

$$\boldsymbol{o_l} = a(W_F^T * \boldsymbol{o}_{l-1} + \boldsymbol{b}) \tag{2.13}$$

schreiben.

Aus der Kombination der jeweiligen Schichten ergibt sich die Architektur des Netzwerkes. Der Nomenklatur aus [31] folgend wird in dieser Arbeit mit Architektur ausgedrückt, wie viele Neuronen ein Netzwerk hat, wie diese Neuronen in Schichten angeordnet sind und wie diese Schichten miteinander verbunden sind.

In dieser Arbeit werden Feed-Forward-Netzwerke[32] verwendet. Mehrere Schichten werden hintereinander angeordnet. Der Ausgang von einer Schicht ist der Eingang der nächsten Schicht[33]. Die Gesamt-Übertragungsfunktion $o_{\text{ges}}$ für solche ein Netzwerk mit $n$ Schichten ergibt sich aus der Verkettung von Formel 2.4 zu:

$$o_{\text{ges}} = o_L(o_{L-1}(\dots o_2(o_1(i, b)) \dots)), \tag{2.14}$$

wo bei $o_l$ der Index $l$ für die $l$-te Schicht des Netzwerkes steht. Die Anzahl an Schichten, also $L$ in Gleichung 2.14, wird als Tiefe des Netzwerkes bezeichnet. Die Breite des Netzwerkes korrespondiert in einer unpräzisen

32 Auf Netzwerke mit zusätzlichen Feedback-Verbindungen, zum Beispiel *Recurrent Neural Network* (RNN) wird im Rahmen dieser Arbeit nicht näher eingegangen

33 In vielen modernen Netzwerken gibt es nicht nur die hier beschriebenen Verbindungen von einer Schicht zur direkt nachfolgenden Schicht. Zusätzlich werden Verbindungen verwendet, bei denen einzelne Schichten übersprungen werden, engl. *skip connections*. Hierdurch können beim nachfolgend beschriebenen Backpropagation-Algorithmus Gradienten besser vom Ausgang des Netzwerkes zu den Schichten nahe dem Eingang des Netzwerkes propagiert werden.

Definition zur Größe der jeweiligen Schichten. Die genaue Wahl der Netzwerktiefe und -breite bedarf der experimentellen Bestimmung.

Faltungsnetzwerke, engl. CNN, bestehen in der Regel aus mehreren Faltungsschichten gefolgt von einer oder mehreren vollverbundenen Schichten. In der ersten Faltungsschicht werden einfache Merkmale wie Kanten extrahiert, welche dann mit zunehmender Tiefe des Netzwerkes abstrahiert werden bis am Ausgang des Netzwerkes beispielsweise Objekte erkannt werden.

Das *Universal Approximation Theorem*, dessen vertiefte Diskussion den Rahmen dieser Arbeit sprengen würde, besagt vereinfacht, dass neuronale Netzwerke mit einer hinreichenden Tiefe und Breite und einer geeigneten Aktivierungsfunktion jede Borel-messbare Funktion, hierzu zählen alle stetigen Funktionen, mit einem beliebig kleinen Fehler darstellen können [31, 42, 73][34].

*Training neuronaler Netzwerke*

Das Ziel des Trainings neuronaler Netzwerke ist, dass die Übertragungsfunktion des Netzwerks, $o_{\text{ges}}$ in Gleichung 2.14, durch die Anpassung der Netzwerkparameter die Funktion der zu lernenden Aufgabe $T$ (vgl. Seite 28) möglichst gut annähert.

Beim Training neuronaler Netzwerke wird die von den Parametern des Netzwerkes, zum Beispiel von den Gewichten und Bias-Werten, abhängende Kostenfunktion, welche die Leistung des Netzwerkes für die betrachtete Aufgabe misst, minimiert. Für die Minimierung werden Gradientenabstiegsverfahren [14] eingesetzt. Hierbei wird in einem iterativen Prozess die Kostenfunktion minimiert, indem die die Kostenfunktion beeinflussenden Parameter in kleinen Schritten entgegen des Gradienten der Kostenfunktion nach ebenjenen Parametern angepasst werden[35]. Durch die kleinen Schritte in Richtung kleinerer Werte für die Kostenfunktion soll im Idealfall das globale Minimum der Kostenfunktion erreicht werden. Im Kontext neuronaler Netzwerke wird das Erreichen des globalen Minimums durch zahlreiche lokale Minima, Sattelpunkte und flache Bereiche in der Kostenfunktion erschwert. Hierbei werden die Gradienten zu klein um ein Weiterlernen zu ermöglichen, engl. *vanishing*

34 Das Theorem besagt lediglich, dass solch ein Netzwerk existiert. Es lässt jedoch keine Schlüsse über die tatsächliche Größe und Architektur oder über das notwendige Vorgehen beim Training des Netzwerkes zu. Insbesondere kann ein als universeller Approximator geeignetes Netzwerk so groß werden, dass ein Erlernen der zu lernenden Funktionen in der Praxis unmöglich ist.

35 Der Gradient der Kostenfunktion in einem Arbeitspunkt zeigt in Richtung der größten Steigung der Kostenfunktion in diesem Arbeitspunkt. Deshalb erfolgt der Schritt entgegen des Gradienten, da in dieser Richtung die Verrringerung des Wertes der Kostenfunktion am größten ist. Dieses entspricht dem größten Fortschritt beim Training.

*gradients*. Zu groß werdende Gradienten, engl. *exploding gradients*, können ebenfalls zu einem Problem werden, insbesondere in tiefen Netzwerken in denen viele Gewichte in einer Kette miteinander multipliziert werden.

In dem nachfolgenden Abschnitt (ab Seite 43) zum Back-Propagation-Algorithmus wird der Gradient konkret in Abhängigkeit der Gewichte und Bias-Werte bestimmt werden. An dieser Stelle sei die Konfiguration des Netzwerkes, inklusive Gewichten und Bias-Werten, zunächst allgemein als $\boldsymbol{c}$ zusammengefasst. Die Kostenfunktion in Abhängigkeit dieser Konfiguration sei $C(\boldsymbol{c})$. Der Gradient der Kostenfunktion nach der Konfiguration sei $\nabla_{\boldsymbol{c}} C(\boldsymbol{c})$. Dann ergibt sich die neue Konfiguration des Netzwerkes, $\boldsymbol{c}_{i+1}$, nach einem Schritt $i$ mit der Weite $\eta$ aus der alten Konfiguration $\boldsymbol{c}_i$ zu:

$$\boldsymbol{c}_{i+1} = \boldsymbol{c}_i - \eta \nabla_{\boldsymbol{c}_i} C(\boldsymbol{c}_i) \tag{2.15}$$

Die Schrittweite $\eta$ wird im Kontext maschineller Lernverfahren auch als Lernrate bezeichnet[36].

Typischerweise werden für maschinelle Lernalgorithmen Datensätze mit vielen Trainingsbeispielen verwendet. Die Berechnung des Gradienten ergibt sich dann als Mittelung über je einen Gradienten pro Beispiel:

$$\nabla_{\boldsymbol{c}} C(\boldsymbol{c}) = \frac{1}{N} \sum_{n=1}^{N} \nabla_{\boldsymbol{c}} C_n(\boldsymbol{c}), \tag{2.16}$$

mit dem Index $n \in [1, N]$ für die einzelnen Beispiele.

Die Komplexität der Berechnung in Gleichung 2.16 ist $\mathcal{O}(N)$ bei einem Trainingsdatensatz der Größe $N$. Da diese Berechnung in jeder Iteration des Gradientenabstiegs durchgeführt werden müsste und Datensätze oft viele Millionen oder Milliarden Beispiele enthalten, ist diese Berechnung in der Praxis nicht umsetzbar.

Statt des bis hierhin beschriebenen ursprünglichen Gradientenabstiegs wird deshalb in der Praxis der stochastische Gradientenabstieg eingesetzt. Hierbei wird für jede Iteration eine zufällige Untermenge an Trainingsbeispielen, in der Regel maximal eine dreistellige Anzahl, notiert als $N^*$ mit $N^* << N$ aus der Gesamtmenge an Trainingsbeispielen gezogen[37]. Die Prämisse hierbei ist, dass mit dieser Untermenge, welche auch

36 In der Regel wird die Lernrate während des Trainings laufend angepasst. Goodfellow et al. bezeichnen diese Anpassung „mehr als Kunst denn als Wissenschaft“und führen weiter aus, das die meisten allgemeinen Richtlinien zur Anpassung der Lernrate mit Vorsicht zu betrachten seien [31].

37 Bei sehr großen Datensätzen erfolgt aus Rechenaufwandsgründen in der Regel kein zufälliges Ziehen von Beispielen in jeder Iteration. Stattdessen wird der komplette Datensatz einmalig zufällig durchgemischt und dann während des Trainings sequenziell verarbeitet.

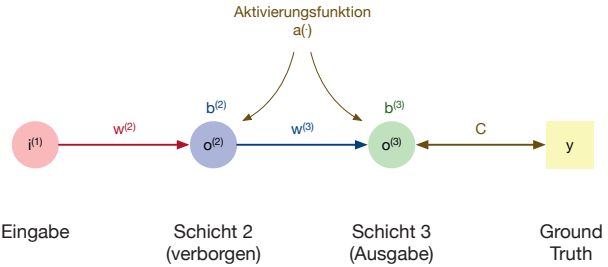



Abbildung 2.9: Backpropagation-Algorithmus für ein sehr kleines neuronales Netzwerk

*Minibatch*[38] bezeichnet wird, der Gradient hinreichend gut angenähert wird:

$$\nabla_c C(c) = \frac{1}{N} \sum_{n=1}^{N} \nabla_c C_n(c) \approx \frac{1}{N^*} \sum_{n=1}^{N^*} \nabla_c C_n(c) \tag{2.17}$$

Diese Annahme wird nicht zuletzt dadurch bestärkt, dass große Datensätze typischerweise nicht redundanzfrei sind [31].

Eine größere (Mini-) Batchgröße führt typischerweise zu einer besseren Annäherung des Gradienten in Gleichung 2.17 und zu einer besseren Auslastung der Rechnerarchitektur, da alle Beispiele aus dem Minibatch in der Regel parallel verarbeitet werden. Kleinere Batchgrößen führen oft zu einem kleineren Generalisierungsfehler, da weniger Rauschen durch die Mittelung entfernt wird, jedoch auch zu längeren Trainingsdauern. Die obere Grenze des Minibatches ergibt sich aus dem in der Rechnerarchitektur zur Verfügung stehenden Arbeitsspeicher. [31]

Da die direkte Berechnung des Gradienten in Gleichung 2.17 für Netzwerke mit vielen Parametern weiterhin sehr rechenaufwendig wäre, wird hierfür der Backpropagation-Algorithmus verwendet.

Zunächst wird der für die Berechnung des Gradienten der Kostenfunktion verwendete Backpropagation-Algorithmus für ein sehr kleines Netzwerk gemäß Abbildung 2.9 mit drei Schichten, also einer verborgenen Schicht, mit jeweils einem Neuron erläutert. Anschließend erfolgt eine Verallgemeinerung auf größere Netzwerke. Das Grundprinzip des Backpropagation-Algorithmus ist die Propagierung des Fehlers am Ausgang des Netzwerkes, gemessen mit der Kostenfunktion $C$, von Schicht zu Schicht durch das Netzwerk um so die Gewichte mittels des stochastischen Gradientenabstiegsverfahrens anzupassen. Durch dieses Vorgehen

38 Es ist zu beachten, dass in der Literatur zum Teil der Begriff *Batch Learning* verwendet wird, um das Lernen auf dem gesamten Datensatz zu bezeichnen. Dieses mag verwirren, da die Größe des Minibatches auch als *Batchgröße* bezeichnet wird.

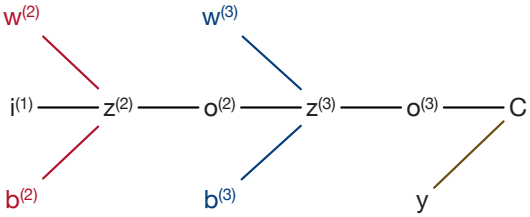



Abbildung 2.10: Abhängigkeiten beim Backpropagation-Algorithmus für ein sehr kleines neuronales Netzwerk

muss lediglich jeweils der Gradient für eine Schicht aber nicht der Gradient für alle Schichten zusammen berechnet werden.

Entsprechend Gleichung 2.4 ergibt sich der Ausgang $o_n^{(3)}$ des Neurons der letzten Schicht, Superscript (3)[39], für das Trainingsbeispiel $n$ zu

$$o_n^{(3)} = a(w^{(3)} o_n^{(2)} + b^{(3)}) \tag{2.18}$$

$$= a(w^{(3)} a(w^{(2)} i_n^{(1)} + b^{(2)}) + b^{(3)}). \tag{2.19}$$

Der Vollständigkeit halber wird auch das Eingangssignal $i_n^{(1)}$ mit dem entsprechenden Schicht-Index notiert, da nicht notwendigerweise alle Eingangssignale in der ersten Schicht in das Netzwerk gehen müssen. Zur verkürzten Notation wird der innere Term von Gleichung 2.18 als $z_n^{(3)}$ definiert:

$$z_n^{(3)} := w^{(3)} o_n^{(2)} + b^{(3)} \tag{2.20}$$

Zur Vereinfachung sei als Kostenfunktion am Ausgang des Netzwerkes der quadratische Fehler zwischen dem Ausgangssignal des Netzwerks und den Trainingsdaten $y$ angenommen:

$$C_n = (o_n^{(3)} - y)^2 \tag{2.21}$$

Anhand von Abbildung 2.10 lässt sich der Weg des Fehlers durch das Netzwerk und damit die Abhängigkeiten der Kostenfunktion von den verschiedenen Werten in dem Netzwerk nachvollziehen: $C$ hängt von $o^{(3)}$ ab (Term *Abh. a* in Gleichung 2.22), $o^{(3)}$ wiederum hängt von $z^{(3)}$ ab (*Abh. b* in Gleichung 2.22) und weiter in der Kette hängt $z^{(3)}$ von $w^{(3)}$ (*Abh. c* in Gleichung 2.22), $o^{(2)}$ und $b^{(3)}$ ab.[40]

39 Der Superscript mit dem Index für die Schicht wird in Klammern gesetzt, um eine Verwechselung mit einer Potenzierung zu vermeiden.

40 Eine umgangssprachliche, häufig referenzierte, Formulierung ist: Wie verändert sich der Wert der Kostenfunktion, wenn an den Gewichten „leicht gewackelt wird"?

Diese Abhängigkeiten lassen sich mittels Kettenregel wie folgt formulieren:

$$\frac{\partial C_n}{\partial w^{(3)}} = \underbrace{\frac{\partial z_n^{(3)}}{\partial w^{(3)}}}_{\text{Abh. c}} \underbrace{\frac{\partial o_n^{(3)}}{\partial z_n^{(3)}}}_{\text{Abh. b}} \underbrace{\frac{\partial C_n^{(3)}}{\partial o_n^{(3)}}}_{\text{Abh. a}} \tag{2.22}$$

Die einzelnen Terme in Gleichung 2.22 ergeben sich zu:

$$\frac{\partial z_n^{(3)}}{\partial w^{(3)}} = o_n^{(2)} \qquad \text{mit Gleichung 2.20} \tag{2.23}$$

$$\frac{\partial o_n^{(3)}}{\partial z_n^{(3)}} = a'(z_n^{(3)}) \tag{2.24}$$

$$\frac{\partial C_n^{(3)}}{\partial o_n^{(3)}} = 2(o_n^{(3)} - y_n) \qquad \text{mit Gleichung 2.21,} \tag{2.25}$$

mit $a'$ als Ableitung der Aktivierungsfunktion. Also ist

$$\frac{\partial C_n}{\partial w^{(3)}} = o_n^{(2)} a'(z_n^{(3)}) 2(o_n^{(3)} - y_n). \tag{2.26}$$

Analog lassen sich die Ableitungen der Kostenfunktion nach dem Bias-Wert und dem Ausgang des vorherigen Neurons berechnen:

$$\frac{\partial C_n}{\partial b^{(3)}} = \frac{\partial z_n^{(3)}}{\partial b^{(3)}} \frac{\partial o_n^{(3)}}{\partial z_n^{(3)}} \frac{\partial C_n^{(3)}}{\partial o_n^{(3)}} \tag{2.27}$$

$$= 1\, a'(z^{(3)})\, 2(o_n^{(3)} - y_n) \tag{2.28}$$

und

$$\frac{\partial C_n}{\partial o_n^{(2)}} = \frac{\partial z_n^{(3)}}{\partial o_n^{(2)}} \frac{\partial o_n^{(3)}}{\partial z_n^{(3)}} \frac{\partial C_n^{(3)}}{\partial o_n^{(3)}} \tag{2.29}$$

$$= w^{(3)}\, a'(z^{(3)})\, 2(o_n^{(3)} - y_n). \tag{2.30}$$

Bei der Betrachtung von $N^*$ Trainingsbeispielen – $N^*$ entspricht zum Beispiel der Batch-Größe – ergibt sich durch Mittelung aus Gleichung 2.26:

$$\frac{\partial C}{\partial w^{(3)}} = \frac{1}{N^*} \sum_{n=1}^{N^*} \frac{\partial C_n}{\partial w^{(3)}} \tag{2.31}$$

$$= \frac{1}{N^*} \sum_{n=1}^{N^*} o_n^{(2)} a'(z_n^{(3)}) 2(o_n^{(3)} - y_n) \tag{2.32}$$

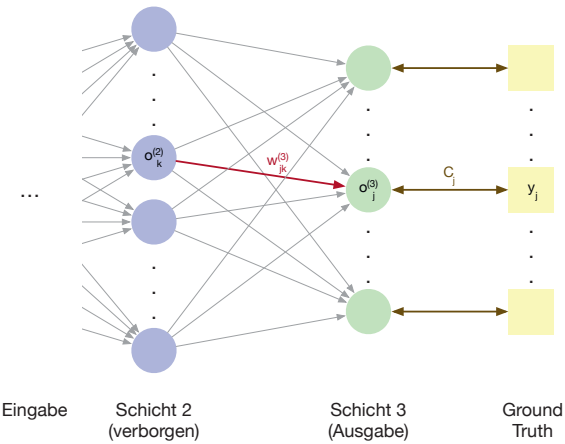



Abbildung 2.11: Backpropagation-Algorithmus für ein neuronales Netzwerk mit einer verborgenen Schicht und mehreren Neuronen pro Schicht

Entsprechende Berechnungen lassen sich auch für die beiden anderen betrachteten Ableitungen durchführen.

Es ist zu beachten, dass während des Trainings lediglich die Gewichte $w$ und die Bias-Werte $b$, nicht jedoch die Ausgangswerte vorheriger Neuronen, direkt beeinflusst werden können. Letztere tauchen jedoch noch in den bis hier hergeleiteten Termen auf. Durch weiteres Anwenden der Kettenregel lassen sich diese nicht direkt beeinflussbaren Terme eliminieren, sodass die Veränderung der Kostenfunktion nur noch von Veränderungen der beeinflussbaren, also der trainierbaren, Terme abhängt.

Die vorherigen Gleichungen für ein neuronales Netzwerk mit nur einem Neuron pro Schicht lassen sich wie im Folgenden ausgeführt auf ein neuronales Netzwerk gleicher Tiefe mit mehreren Neuronen pro Schicht, entsprechend Abbildung 2.11, durch die Ergänzung von weiteren Indizes verallgemeinern. Die Neuronen in Schicht 2 werden mit $k \in [1, K]$ indiziert, die Neuronen in Schicht 3 mit $j \in [1, J]$. Gewichte erhalten entsprechend zwei Indizes, $w_{jk}$, da sie zwei Neuronen mit den zugehörigen Indizes $k$ und $j$ miteinander verknüpfen.

Die Kostenfunktion am Ausgang des Netzwerkes ergibt sich dann als Summierung über die Kosten für die einzelnen Neuronen in der letzten Schicht zu

$$C_n = \sum_{j=1}^{J} (o_{j,n}^{(3)} - y_j)^2, \tag{2.33}$$

mit

$$o_{j,n} = a(z_{j,n}^{(3)}) \tag{2.34}$$

und

$$z_{j,n}^{(3)} = w_{j0}^{3} o_{0,n-1}^{(2)} + \ldots + w_{jk}^{3} o_{k,n-1}^{(2)} + \ldots + w_{jK}^{3} o_{K,n-1}^{(2)} + b_j^{(3)}. \tag{2.35}$$

Die Ableitungen der Kostenfunktion für ein Beispiel $n$ nach den Gewichten und den Bias-Werten ergeben sich durch Ergänzung der entsprechenden Indizes in den Gleichungen 2.22 und 2.27. Im Folgenden wird die Ergänzung nur für die Gewichte ausformuliert. Die zweite Gleichung ergibt sich entsprechend.

$$\frac{\partial C_n}{\partial w_{jk}^{(3)}} = \frac{\partial z_{j,n}^{(3)}}{\partial w_{jk}^{(3)}} \frac{\partial o_{j,n}^{(3)}}{\partial z_{j,n}^{(3)}} \frac{\partial C_n^{(3)}}{\partial o_{j,n}^{(3)}}. \tag{2.36}$$

Bei der Ableitung nach den Ausgangswerten vorheriger Neuronen wird in Gleichung 2.29 zusätzlich zu den Indizes eine Summe ergänzt, da jedes vorherige Neuron in einem vollverbundenen Netzwerk alle Neuronen der nachfolgenden Schicht beeinflusst:

$$\frac{\partial C_n}{\partial o_{k,n}^{(2)}} = \sum_{j=1}^{J} \frac{\partial z_{j,n}^{(3)}}{\partial o_{k,n}^{(3)}} \frac{\partial o_{j,n}^{(3)}}{\partial z_{j,n}^{(3)}} \frac{\partial C_n^{(3)}}{\partial o_{j,n}^{(3)}}. \tag{2.37}$$

Durch Hinzufügen weiterer Indizes und Summen lässt sich die Gleichung für ein neuronales Netzwerk mit einer beliebigen Anzahl an Schichten erweitern. Da sich hierdurch kein neuer Erkenntnisgewinn ergibt wird an dieser Stelle zur Wahrung der Übersichtlichkeit auf die Erweiterung verzichtet.

Diese Gradienten werden für das zuvor beschriebene Gradientenabstiegsverfahren verwendet.

Bei einer hohen Varianz zwischen den Gradienten in aufeinanderfolgenden Iterationen des Gradientenabstiegsverfahrens ist der Lernvorgang nicht sehr zielgerichtet. Zur Umgehung dieses Problems wird der Aktualisierungsschritt aus Gleichung 2.15 häufig um ein Momentum ergänzt. Hierbei wird für den Aktualisierungsschritt nicht nur der in der aktuellen Iteration berechnete Gradient verwendet, sondern eine Akkumulation der vorherigen Gradienten mit exponentiell abfallender Gewichtung. Die Aktualisierungsschrittweite $\alpha_i$ in Iteration $i$ wird definiert als:

$$\alpha_i = \mu \alpha_{i-1} - \eta \nabla_{\boldsymbol{c_i}} C(\boldsymbol{c_i}) \tag{2.38}$$

mit $\mu$ als Momentum-Hyperparameter. Um den exponentiellen Abfall der Gewichtung vergangener Gradienten zu erreichen ist $\mu \in [0, 1)$. Oft

wird $\mu = 0.9$ als Wert für diesen Hyperparameter gewählt. Der Aktualisierungsschritt in Gleichung 2.15 wird dann zur folgenden Formulierung geändert:

$$\boldsymbol{c}_{i+1} = \boldsymbol{c}_i + \alpha_i \tag{2.39}$$

*Rahmenbedingungen für den Entwurf neuronaler Netzwerke*

Ähnlich wie bei der zuvor erläuterten Encodersteuerung für Videocodecs gibt es auch für den Entwurf neuronaler Netzwerke Rahmenbedingungen, innerhalb derer der Entwurf erfolgt. Dieser Rahmen wird durch die zur Verfügung stehende Menge an Trainingsdaten, durch die vertretbare Trainingsdauer, durch Grenzen für den Rechenaufwand der Inferenz sowie durch den Speicherbedarf des Netzwerkes gesetzt.

Die Menge an verfügbaren Trainingsdaten ist für verschiedene Anwendungen sehr unterschiedlich. Für manche Anwendungen gibt es sehr viele Daten. Dieses gilt zum Beispiel für sehr stark beforschte Themengebiete wie der Bildklassifikation oder für Anwendungen, in denen sich synthetische Daten automatisiert erstellen lassen. In anderen Anwendungsgebieten ist die Menge stark begrenzt. Das ist zum Beispiel der Fall, wenn Daten händisch verarbeitet werden müssen bevor sie als Trainingsdaten verwendbar sind. Für das Training großer Netzwerke werden in der Regel sehr viele Daten benötigt. Das Training von neuronalen Netzwerken mit sehr wenigen Trainingsbeispielen ist Gegenstand der aktuellen Forschung [101].

Die Trainingsdauer für neuronale Netzwerke und insbesondere der Rechenaufwand für die Optimierung der Hyperparameter kann erheblich sein. So berichten Zoph und Le von einem Experiment [151], bei dem zur Hyperparameter-Optimierung für mehrere Wochen auf 800 Grafikprozessoren, engl. Graphics Processing Unit (GPU), gleichzeitig gerechnet wurde, um eine Verbesserung des Fehlers von 0,09% bei einer Beschleunigung um einem Faktor von 1,05 gegenüber dem zuvor besten Modell zu erreichen. In Abhängigkeit der zur Verfügung stehenden Ressourcen kann der vertretbare Rechenaufwand variieren.

In der Literatur werden in der Regel höchstens Angaben zur Anzahl an verwendeten GPUs oder Tensor Processing Unit (TPU)s sowie der Trainingsdauer gemacht – beides lediglich für das final verwendete Modell. Das Training findet typischerweise auf den Servern von Cloud-Anbietern wie Google oder Amazon statt. Durch den Abgleich der Preislisten zur Anmietung der Cloud-Rechenkapazitäten mit den Trainingsdauer- und Ressourcenverwendungs-Angaben schätzen Sharir et al. in [110] die finanziellen Kosten für das Training ab: Für verschiedene Netzwerk-Architekturen aus dem Stand der Technik im Bereich der

Sprachverarbeitung schätzen sie die Trainingskosten für einen einmaligen Durchlauf des Trainings auf Werte zwischen $2.500 (Netzwerk mit 110 Millionen Parametern) bis $80.000 (für 1,5 Milliarden Parameter, das aktuell größte veröffentlichte Netzwerk). Unter der Berücksichtigung der Tatsache, dass für Veröffentlichungen häufig nicht nur ein einmaliges Training durchgeführt wird, sondern mehrere Architekturen ausprobiert und Hyperparameter optimiert werden, kommen die Autoren inklusive dieser versteckten Kosten auf Trainingskosten für die entsprechenden Netzwerke zwischen $50.000 und $1.600.000.

Insbesondere für zeitkritische Anwendungen, zum Beispiele solche mit einer Echtzeitanforderung, ist der Rechenaufwand für die Inferenz relevant. Gegebenfalls kann es für manche Anwendungen nur möglich sein, kleine Netzwerke mit vergleichsweise wenigen Rechenoperationen zu entwerfen, da andernfalls keine Anwendbarkeit in Echtzeit mehr möglich ist.

Für Anwendungen, die auf mobilen Endgeräten oder eingebetten Systemen ausgeführt werden, oder bei denen die trainierten Netzwerkmodelle über mobile Datenverbindungen zu übertragen sind, wird die Dateigröße des Modells relevant und kann ein begrenzender Faktor werden. Die Speicherplatzreduktion für neuronale Netzwerke ist ebenfalls Gegenstand der aktuellen Forschung [67].

## 2.3 KONTURDETEKTION

Das in Kapitel 3 vorgeschlagene Verfahren beruht auf der Modellierung und Extrapolation von Konturen. Somit ist die Detektion von Konturen ein notwendiger Vorverarbeitungsschritt.

Eine beispielhafte Darstellung eines eindimensionalen Kontursignals ist in Abbildung 2.12 zusammen mit der ersten und zweiten Ableitung des Signals zu sehen. Prinzipiell lassen sich Konturen als Maximum der ersten Ableitung beziehungsweise als Nulldurchgang in der zweiten Ableitung mit hinreichend großen Ausschlägen direkt davor und danach bestimmen.

Für zweidimensionale Bilder $I$ lässt sich der Gradient $\nabla_I$ mit den partiellen Ableitungen in $x$- und $y$-Richtung berechnen:

$$\nabla_I = \begin{pmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{pmatrix} \tag{2.40}$$

Der Betrag $|\nabla_I|$ des Gradienten berechnet sich wie folgt:

$$|\nabla_I| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2} \tag{2.41}$$

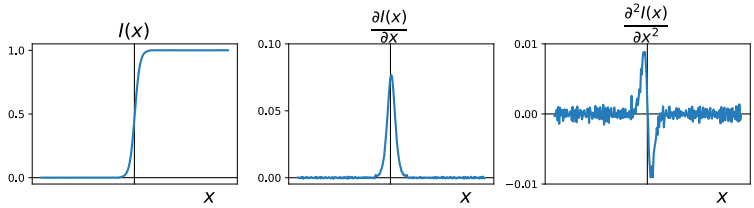



Abbildung 2.12: Darstellung der ersten (Mitte) und zweiten (rechts) Ableitung eines eindimensionalen Signals mit einer Kontur (links).

Durch Maximumsuche in dem Signal nach Gleichung 2.41 ergibt sich die Position der Kontur. Eine entsprechende Berechnung für die zweite Ableitung wird hier nicht näher ausgeführt aber ergibt sich analog.

In der Praxis ist diese Art der Konturdetektion jedoch nicht robust genug für reale Bilder. Stattdessen wird in dieser Arbeit der *Canny*-Konturdetektionsalgorithmus [12] verwendet, welcher im Folgenden näher ausgeführt wird. Der Canny-Algorithmus besteht aus fünf Schritten.

Im ersten Schritt wird das Bildsignal mit einem Tiefpass in Form einer Gauß-Funktion gefiltert. Durch die Filterung soll Rauschen, welches als additiv, weiß und gauß-verteilt angenommen wird, entfernt werden. Das quadratische Filter $T$ mit einer Kantenlänge von $a = 2k + 1$ ist definiert als

$$T = (h_{ij}) \text{ mit } h_{ij} = \frac{1}{2\pi\sigma^2} e^{-\frac{(i-(k+1))^2+(j-(k+1))^2}{2\sigma^2}} \text{ und } i, j \in [1, 2k+1] \quad (2.42)$$

Das gefilterte Bild ergibt sich durch Faltung zu

$$I_{\text{filt}} = T * I \quad (2.43)$$

Im zweiten Schritt werden die Bildgradienten mit dem Sobel-Operator [112] bestimmt. Die Sobel-Operatoren für Konturen in $x$- und $y$-Richtung sind wie folgt definiert:

$$K_{\text{Sobel},x} = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \text{ und } K_{\text{Sobel},y} = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \quad (2.44)$$

Mit diesem Operator findet in Konturrichtung eine Annäherung des Gradienten statt, während orthogonal zur Konturrichtung das Signal geglättet wird. Mittels des Sobel-Operators werden die Gradientenbilder

in beiden Richtungen, $G_x$ und $G_y$, aus dem im ersten Schritt erzeugten gefilterten Bild berechnet:

$$G_x = K_{\text{Sobel},x} * I_{\text{filt}} \text{ und } G_y = K_{\text{Sobel},y} * I_{\text{filt}} \tag{2.45}$$

Aus den beiden Gradientenbildern wird durch jeweils punktweise Berechnung für jeden Abtastwert der Betrag des Gradienten $G$ sowie die Richtung des Gradienten $\Phi$ bestimmt:

$$G = \sqrt{G_x^2 + G_y^2} \tag{2.46}$$

$$\Phi = \text{atan}(G_y, G_x) \tag{2.47}$$

Im dritten Schritt wird das Betragsgradientenbild $G$ aus Gleichung 2.46 mittels *Non-maximum Suppresion* (NMS), der Unterdrückung von Werten, die kein Maximum sind, ausgedünnt. In Folge der Ausdünnung wird das Skelett der Konturen, $G_{\text{skel}}$, ermittelt. Hierfür wird der Wert jedes Pixels in $G$ mit den Werten $n_1$ und $n_2$ der beiden benachbarten Pixel in Richtung und entgegen des Gradienten ($\Phi(x,y)$) an der entsprechenden Stelle verglichen. Nur wenn $G(x,y)$ größer als die beiden Nachbarwerte ist, bleibt der Wert erhalten. Andernfalls wird der Wert auf Null gesetzt. Also:

$$G_{\text{skel}}(x,y) = \begin{cases} \boldsymbol{G}(x,y) & \text{wenn } \boldsymbol{G}(x,y) > n_1, n_2 \\ 0 & \text{sonst.} \end{cases} \tag{2.48}$$

Die verbleibenden Konturpixel in $G_{\text{skel}}$ werden im vierten Schritt mittels zweier Grenzwerte, $\tau_1$ und $\tau_2$ mit $\tau_1 < \tau_2$, in starke Konturpixel, schwache Konturpixel und keine Konturpixel klassifiziert. Die Matrix mit den Klassifikationsergebnissen $\mathcal{K}$ berechnet sich gemäß

$$\mathcal{K}(x,y) = \begin{cases} \text{starker Konturpixel} & \text{wenn } G_{\text{skel}}(x,y) > \tau_2 \\ \text{schwacher Konturpixel} & \text{wenn } \tau_1 < G_{\text{skel}}(x,y) < \tau_2 \\ \text{kein Konturpixel} & \text{sonst.} \end{cases} \tag{2.49}$$

Die als starke Konturpixel klassifizierten Pixel werden als auf jeden Fall zu einer Kontur gehörend betrachtet. Über die als schwache Konturpixel klassifizierten Pixel wird im fünften Schritt mittels einer Hysterese entschieden. Ausgehend von allen starken Konturpixeln werden in beide Richtungen gehend schwache Konturpixel als zur Kontur gehörend

betrachtet, solange sie mit starken Konturpixel oder mit als zur Kontur gehörend entschiedenen schwachen Konturpixeln verbunden sind. Die finale Matrix mit einem binären Konturbild nach der Canny-Konturdetektion wird als $I_{\text{Canny}}$ bezeichnet.

Für CoMIC v1 wurde ermittelt, dass durch die Verwendung von ausgefeilteren Konturdetektionsalgorithmen, zum Beispiel [22], ein minimaler zusätzlicher Codierungsgewinn von 0,3% erzielt werden kann. Da dieser zusätzliche Codierungsgewinn jedoch mit einer Erhöhung der Rechenkomplexität des gesamten Verfahrens um einen Faktor von 90 einhergeht, erfolgt eine Beschränkung auf den schneller berechenbaren Canny-Algorithmus.

# 3

# VERFAHREN ZUR MODELLIERUNG VON KONTUREN

Ausgehend von einem zu codierenden Block, mit dem zugehörigen ihn umgebenden Referenzbereich als Prädiktionsgrundlage, wird in diesem Kapitel die Modellierung und Extrapolation der Konturen mit dem vorgeschlagenen stochastischem Konturmodell beschrieben. Außer Betracht gelassen wird hierbei die Partitionierung des Bildes in Blöcke. Im weiteren Verlauf der Arbeit werden im Ergebniskapitel (Kapitel 5) zwei Implementierungen betrachtet: Zum einen der selbstentwickelte CoMIC-Codec sowie der Videocodec HEVC. In ersterem werden lediglich Blöcke mit stets der gleichen, zu Beginn der Codierung festgelegten, Blockgröße codiert während in letzterem durch die ausgefeiltere Partitionierung Blöcke in verschiedenen Größen entstehen. Im Rest dieses Kapitels werden die Verfahren unter Annahme quadratischer Blöcke diskutiert und visualisiert. Dieses stellt jedoch keine Einschränkung der beschriebenen Methoden dar.

Um die Konsistenz mit den eigenen Vorarbeiten [65, 69] zu wahren wird der Referenzbereich so wie in Abbildung 3.1 visualisiert gewählt: Der Referenzbereich befindet sich in vier Bildbereichen links, oben, links-oben sowie rechts-oben vom aktuell codierten Block. Die vier Bildbereiche, die zusammen den Referenzbereich ergeben, sind jeweils so groß wie der aktuell zu codierende Block. Da ein kausales Codierungssystem betrachtet wird, ist der Bereich rechts neben dem aktuellen Block nicht referenzierbar.

## 3.1 KONTURDETEKTION

Die Konturdetektion im Rahmen dieser Arbeit unterscheidet sich nicht im Vergleich zu den eigenen Vorarbeiten [65, 69]. Sie wird nachfolgend kurz zusammengefasst, damit die vorliegende Arbeit in sich geschlossen ist. Die Konturdetektion wird in den Bereichen des Referenzbereichs durchgeführt, in denen die Abtastwerte verfügbar ist. In Abhängigkeit der gewählten Partitionierung und an den Rändern des Bildes kann es zu Einschränkungen kommen.

Der im vorherigen Kapitel beschriebene Canny-Algorithmus [12] zur Konturdetektion wird auf den Referenzbereich angewendet. Der Canny-

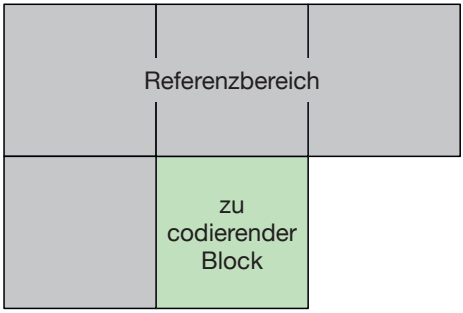



Abbildung 3.1: Darstellung des Referenzbereichs. Die vier Blöcke des Referenzbereichs sind genauso groß wie der zu codierende Block.

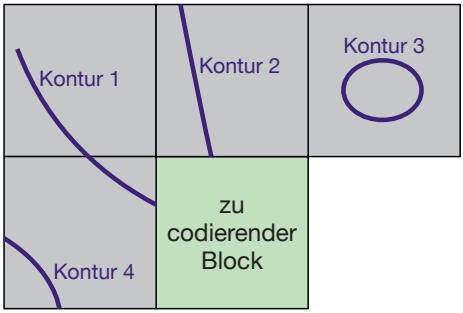



Abbildung 3.2: Konturen im Referenzbereich. Lediglich die Konturen, die an den zu codierenden Block angrenzen, werden für die Prädiktion verwendet. Im Beispiel handelt es sich um die Konturen 1 und 2.

Algorithmus hat zwei Parameter, mit denen die Klassifizierung der Punkte[1] in starke und schwache Konturpunkte kontrolliert wird. Da die betrachteten Videosignale potentiell sehr unterschiedliche Signalcharakteristiken haben können, ist es nicht zielführend, globale Werte für diese Parameter festzulegen. Stattdessen werden die Parameter signaladaptiv bestimmt. Wie bereits in den eigenen Vorarbeiten [65, 69] und gemäß der Arbeit von Fang et al. [26] wird hierfür die Otsu-Methode [96] eingesetzt. Das Ergebnis des Canny-Algorithmus ist ein binäres Konturbild, in dem alle zu Konturen gehörenden Punkte markiert sind. Ein Beispiel ist in Abbildung 3.2 dargestellt. Mit dem von Suzuki und Be vorgeschlagenen Verfahren aus [117] werden zusammenhängende Konturpunkte in dem binären Konturbild detektiert. Zusammenhängende Konturpunkte gehören zu der selben Kontur, nicht zusammenhängende Konturpunkte gehören folglich zu unterschiedlichen Konturen. Durch das gewählte Verfahren kann es passieren, dass einige Punkte mehrfach einer Kontur hinzugefügt werden. Solche redundant vorkommenden Punkte werden bereinigt.

Das Ergebnis der Konturdetektion sind die $x$- und $y$-Koordinaten der Kontur. Für eine Kontur mit $n$ Konturpixeln $p_i = (x_i, y_i)$ mit $i = 1, \ldots, n$ sei:

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n \times 1}, \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^{n \times 1}. \tag{3.1}$$

Für jede Kontur wird geprüft, ob sie an den zu codierenden Block angrenzt. Nur diese Konturen werden wie in Abbildung 3.3 dargestellt für die nachfolgend beschriebene Prädiktion berücksichtigt. Des Weiteren werden nur Konturen mit mehr als drei Pixeln betrachtet. Dieser Wert hat sich in Vorarbeiten als sinnvolle Abwägung zwischen einer hohen Anzahl an verwendeten Konturen und der notwendigen Robustheit der Konturextrapolation erwiesen.

Bei der Modellierung der Konturen wird unterschieden je nachdem, ob die Konturen von links oder von oben auf den zu prädizierenden Block treffen. Bei Konturen, die von links kommend auf den aktuell zu codierenden Block treffen, wird die horizontale Koordinate $x$ als unabhängige Variable und die vertikale Koordinate $y$ als abhängige Variable betrachtet. Für von oben kommend auf den aktuell zu codierenden Block treffende Konturen ist es umgekehrt. Ohne hiermit einhergehende Ein-

1 Der Begriff *Punkt* wird zur Abgrenzung von den zuvor definierten Pixeln gewählt. In der verwendeten Definition besteht ein Punkt aus zwei örtlichen Koordinaten und einem repräsentierten Wert. Bei diesem Wert kann es sich um einen Abtastwert des Bildes, einen binären Wert einer Konturkarte oder einen Gradientenwert handeln.

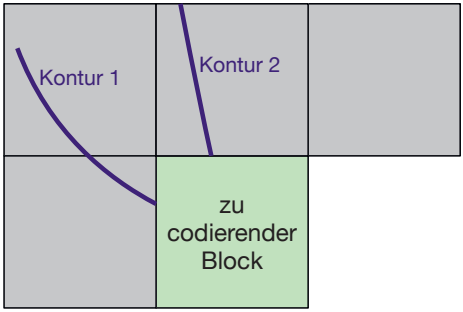



Abbildung 3.3: Relevante Konturen im Referenzbereich. Lediglich die Konturen, die an den zu codierenden Block angrenzen, werden für die Prädiktion verwendet.

schränkungen wird im Folgenden lediglich der Fall betrachtet, dass $x$ die unabhängige Variable ist. Sämtliche Überlegungen und Herleitungen sind analog auf den anderen Fall übertragbar.

## 3.2 KONTURGLÄTTUNG

Durch den gewählten Konturdetektionsalgorithmis sind die Konturpixel bis hierhin nur mit Ganz-Pel-Genauigkeit bekannt. Diese Kontur wird als diskrete Kontur bezeichnet. Hierdurch kommt es zu Instabilitäten bei der Modellierung der Kontur. Um dieses zu vermeiden wird die Kontur zunächst durch eine kontinuierliche Funktion angenähert. Dieses kann auch als Glättung der Kontur betrachtet werden. Dieses passt auch dazu, dass die Kontur in der Realität einen kontinuierlichen Verlauf hatte, welcher nun durch diskrete Koordinaten angenähert wurde.

Für die Annäherung der Kontur werden jeweils vier Funktionen (Polynome nullten, ersten und zweiten Grades, sowie eine Exponentialfunktion) mit der Methode der kleinsten Quadrate bezüglich der diskreten Kontur optimiert. Die Funktion mit dem kleinsten mittleren quadratischen Fehler wird ausgewählt. Der mittlere quadratische Fehler wird berechnet gemäß:

$$\mathrm{MSE}_{\mathrm{Glättung}} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \widetilde{y}_i)^2 \tag{3.2}$$

für die durch die kontinuierliche Funktion angenäherten Werte $\widetilde{y}_i$.

Hierdurch sind die Koordinaten der Konturpixel nun mit Sub-Pel-Genauigkeit gegeben. Diese Kontur-Repräsentation wird als kontinuierliche Kontur bezeichnet.

Die beschriebene Annäherung ist nur sinnvoll, wenn der hierdurch entstehende Fehler nicht zu groß wird. Deshalb wird die Annäherung nur verwendet, wenn der mittlere quadratische Fehler zwischen der kontinuierlichen und der diskreten Kontur kleiner als 1 Pel ist. In diesem Fall ist die Annäherung fast exakt, da der Fehler im Mittel der Genauigkeit der diskreten Kontur entspricht. Für den Fall, dass die Approximation erfolgreich ist, wird im weiteren Verlauf ein rauschfreier Gauß-Prozess modelliert. Falls die Approximation nicht erfolgreich ist, wird die Annahme getroffen, dass die diskrete Kontur durch additives, weißes Rauschen aus der kontinuierlichen Kontur entstanden ist, und der Gauß-Prozess entsprechend für eine rauschbehaftete Kontur formuliert.

## 3.3 MODELLIERUNG DER KONTUR

Der Modellierung der Kontur wird die Prämisse zugrunde gelegt, dass es sich bei den Konturpixeln um Beobachtungen eines stochastischen Prozesses handelt. Ferner wird angenommen, dass die Beobachtungen normalverteilt sind. Für die Modellierung wird ein Gauß-Prozess gewählt. Die Nomenklatur für die Beschreibung der Gauß-Prozesse folgt dem Lehrbuch [100] von Rasmussen und Williams.

Zunächst wird der sogenannte *A-Priori-Gauß-Prozess*, kurz *Prior*, formuliert. In den Prior gehen die beim Entwurf des Gauß-Prozesses getroffenen Annahmen über den zugrundeliegenden Prozess, hier also die Konturen, in Form einer Erwartungswertfunktion und einer Kovarianzfunktion ein. Der Prior ist somit unabhängig von den beobachteten Konturen. Die Wahl von einer geeigneten Erwartungswertfunktion und einer geeigneten Kovarianzfunktion bilden folglich den Kern des Prior-Entwurfs. Die typischerweise gewählten Funktionen haben in der Regel Hyperparameter. Diese bleiben zunächst für den Prior unbestimmt. Basierend auf dem Prior kann für eine konkrete detektierte Kontur der sogenannte *Posterior-Gauß-Prozess*, kurz *Posterior*, bestimmt werden. Hierfür wird der Prior basierend auf den beobachteten Konturpunkten angepasst. Zu diesem Zweck werden die Werte der Hyperparameter der Erwartungswertfunktion sowie der Kovarianzfunktion optimiert. Das Ziel der Optimierung ist das Erreichen einer größtmöglichen Übereinstimmung zwischen den aus dem Posterior hervorgehenden Beobachtungen und den tatsächlich beobachteten Konturpunkten. In den nachfolgenden Abschnitten werden die Bestimmung des Priors und des Posteriors genauer betrachtet.

## 3.4 A-PRIORI-GAUSS-PROZESS

Die Beobachtungen $\mathbf{b}$ des zu modellierenden Prozesses seien Konturpunkte:

$$\mathbf{b} = (\mathbf{x}, \mathbf{y}) \tag{3.3}$$

Die Auswertung des Gauß-Prozesses an den $\mathbf{x}$-Werten der detektierten Kontur liefert die Prior-Auswertungen $\mathbf{f}$. Diese Prior-Auswertungen setzen sich aus den rauschfreien Auswertungen $\bar{\mathbf{f}}$ und unkorreliertem weißen Rauschen $\epsilon$ mit der Varianz $\sigma_n^2$ zusammen:

$$\mathbf{f} = \bar{\mathbf{f}} + \epsilon. \tag{3.4}$$

Für einen rauschfreien Gauß-Prozess ergeben sich die Beobachtungen zu

$$\widetilde{\mathbf{b}} = (\mathbf{x}, \bar{\mathbf{f}}), \tag{3.5}$$

da $\sigma_n^2 = 0$ gesetzt wird. Für einen rauschbehafteten Gauß-Prozess ergibt sich:

$$\widetilde{\mathbf{b}} = (\mathbf{x}, \mathbf{f}) \text{ mit } \mathbf{f} \sim \mathcal{N}\left(0, K(\mathbf{x}, \mathbf{x}) + \sigma_n^2 \mathcal{I}\right). \tag{3.6}$$

Hierbei ist $K$ die Kovarianzmatrix, die sich aus $\mathbf{x}$ ergibt, und $\mathcal{I}$ die Einheitsmatrix.

Aus Gleichung 3.6 lassen sich zwei Beobachtungen ableiten: Die Auswertungen $\mathbf{f}$ an den Stellen $\mathbf{x}$ sind normalverteilt und sie hängen von den Eingaben $\mathbf{x}$ ab. Aufgrund dessen wird $\mathbf{f}$ als bedingte Zufallsvariable $\mathbf{f}|\mathbf{x}$ formuliert. Nachfolgend wird der rauschbehaftete Gauß-Prozess betrachtet. Der rauschfreie Fall lässt sich durch Setzen von $\sigma_n^2 = 0$ herleiten.

Rasmussen und Williams [100] folgend wird, wie bereits in Gleichung 3.6 angedeutet, die Erwartungswertfunktion des Priors zu Null gesetzt. Dieses stellt keine weitreichende Einschränkung dar, sondern dient vielmehr der vereinfachten Notation, da sich keine ebensolche Restriktion für den im weiteren Verlauf der Ausführungen betrachteten Posterior ergibt.

Die Kovarianzfunktion, welche auch als Kernel bezeichnet wird, ist für den Prior von weitaus größerer Bedeutung als die Erwartungswertfunktion. Der sogenannte *Squared Exponential Kernel*, $k_{\mathrm{SE}}(x_p, x_q)$, ist der Defacto-Standard für Gauß-Prozess-Kovarianzfunktionen [25]. Deshalb wird diese Kovarianzfunktion auch hier verwendet. Die Definition lautet:

$$k_{\mathrm{SE}}(x_p, x_q) = \zeta^2 e^{-\frac{(x_p - x_q)^2}{2\zeta^2}} \tag{3.7}$$

Die Variable $\zeta$ ist hierbei ein Skalierungsfaktor. Er wird quadriert notiert, da er auch als Varianz interpretiert wird, die angibt, wie die mittlere Abweichung der zu modellierenden Funktion von ihrem Mittelwert ist.

Der Längenskalierungsfaktor $\xi$ beschreibt die zu erwartende Krümmung der zu modellierenden Kontur.

Die gewählte Kovarianzfunktion bietet den Vorteil, dass sie beliebig oft differenzierbar ist und damit einen glatten Verlauf liefert. Diese Eigenschaft passt dazu, dass für die beobachteten Konturen in vielen Fällen keine Knicke innerhalb der betrachteten Blockgrößen erwartet werden. Dieses geht einher mit den vielfältigen Partitionierungsoptionen, die bei der Codierung in einem Videocodec möglich sind.

Alternativ könnte auch der sogenannte *Rational Quadratic Kernel* verwendet werden. Hierbei handelt es sich um eine Verallgemeinerung des *Squared Exponential Kernels* mit einem zusätzlichen Längenskalierungsfaktor als Hyperparameter. Durch den zusätzlichen Hyperparameter ließen sich Daten, die entlang mehrerer Längenskalen variieren, besser modellieren. Dieses wird jedoch ebenfalls aufgrund der Partitionierungsalgorithmen nicht erwartet.

Ein Nachteil aufgrund der gewählten Kovarianzfunktion ergibt sich, falls Knicke in der modellierten Kontur vorkommen. In diesem Fall passt das gewählte Modell nicht zu den zu modellierenden Daten. In der eigenen Vorarbeit [69] wurde ein Lösungsansatz für dieses Problem vorgeschlagen. Da der Fall jedoch nur selten eintritt, wird er hier nicht weiter betrachtet.

Die gewählte Kovarianzfunktion führt zu einer symmetrischen Kovarianzmatrix. Deshalb lässt sich der Rechenaufwand der nachfolgend beschriebenen Posterior-Bestimmung durch eine Cholesky-Zerlegung verringern.

## 3.5 POSTERIOR-GAUSS-PROZESS

Der Posterior ergibt sich aus dem Prior durch Optimierung der Hyperparameter $\Theta = (\zeta, \xi)$ der Kovarianzfunktion. Die Startwerte der Hyperparameter des Gauß-Prozesses seien $\Theta_0$. Das Optimierungsziel ist hierbei, die Übereinstimmung der Gauß-Prozess-Auswertungen $\mathbf{f}$ mit den tatsächlichen Werten $\mathbf{y}$ der Konturpixel an den Stellen $\mathbf{x}$ zu maximieren. Als Gütekriterium wird die Likelihood verwendet. Auf den Anwendungsfall der Konturmodellierung bezogen handelt es sich um die Wahrscheinlichkeit, dass die Beobachtungen $\mathbf{f}|\mathbf{x}$ des Gauß-Prozesses, die Werte $\mathbf{y}$ der Konturpixel haben. Die Formulierung erfolgt über die Verbundwahrscheinlichkeit $P(\mathbf{y}|\mathbf{f}, \mathbf{x})$ der bedingten Zufallsvariable $\mathbf{y}|\mathbf{f}$ und $\mathbf{x}$.

Die Wahrscheinlichkeit, dass $\mathbf{y}$ unter der Bedingung $\mathbf{x}$ beobachtet wird, ergibt sich als Randwahrscheinlichkeit, engl. Marginal Likelihood, von $\mathbf{y}$ über alle $\mathbf{f}$. Hierbei berechnet sich die Wahrscheinlichkeitsdichte aus dem

Produkt der Prior-Wahrscheinlichkeitsdichte, $p(\mathbf{f}|\mathbf{x})$, und der Likelihood-Wahrscheinlichkeitsdichte, $p(\mathbf{y}|\mathbf{f},\mathbf{x})$:

$$p(\mathbf{y}|\mathbf{x}) = p(\mathbf{y}|\mathbf{f},\mathbf{x}) \cdot p(\mathbf{f}|\mathbf{x}). \tag{3.8}$$

Durch Integration lässt sich aus der Wahrscheinlichkeitsdichte die Wahrscheinlichkeit bestimmen:

$$P(\mathbf{y}|\mathbf{x}) = \int p(\mathbf{y}|\mathbf{f},\mathbf{x}) \cdot p(\mathbf{f}|\mathbf{x})\, \mathrm{d}\mathbf{f}. \tag{3.9}$$

Für die Wahrscheinlichkeitsdichtefunktionen wird die n-dimensionale (vgl. Gleichung 3.1) gauß-förmige Wahrscheinlichkeitsdichte eingesetzt. Diese ist wie folgt definiert:

$$p_x(x) = \frac{1}{\sqrt{2\pi^n|\Sigma|}} \cdot e^{-\frac{1}{2}(\mathbf{x}-\mu)\Sigma^{-1}(\mathbf{x}-\mu)} \tag{3.10}$$

In Gleichung 3.10 steht $\Sigma$ für die Kovarianzmatrix, $|\cdot|$ für den Determinatenoperator und $\mu$ für den Erwartungswertvektor.

Folglich wird die Wahrscheinlichkeit $P(\mathbf{y}|\mathbf{x})$ maximal, wenn die Exponenten der Wahrscheinlichkeitsdichten gemäß Gleichung 3.10 maximal werden. Für diese Maximierung wird durch Logarithmieren unter Annahme von $\mu = 0$ und $\Sigma = K + \sigma_n^2\mathcal{I}$ die *Log-Marginal-Likelihood* berechnet:

$$\ln\left(p\left(\mathbf{y}|\mathbf{x}\right)\right) = \ln\left(\frac{1}{\sqrt{2\pi^n|\Sigma|}} \cdot e^{-\frac{1}{2}\mathbf{y}^T\Sigma^{-1}\mathbf{y}}\right) \tag{3.11}$$

$$= \ln\left(\frac{1}{\sqrt{2\pi^n|\Sigma|}}\right) - \frac{1}{2}\mathbf{y}^T\Sigma^{-1}\mathbf{y} \tag{3.12}$$

$$= -\frac{n}{2}\ln\left(2\pi\right) - \frac{1}{2}\ln\left(|\Sigma|\right) - \frac{1}{2}\mathbf{y}^T\Sigma^{-1}\mathbf{y} \tag{3.13}$$

Der erste Term in Gleichung 3.13 ist eine Konstante, der zweite Term wirkt über die Hyperparameter der Kovarianzfunktion regularisierend auf die Modellkomplexität und der dritte Term beschreibt die Güte der Anpassung des Gauß-Prozesses an die detektierte Kontur [100].

Da $\Sigma$ positiv-definit und symmetrisch ist, können $\Sigma^{-1}$ und $|\Sigma|$ mit der in Gleichung 3.14 notierten Cholesky-Zerlegung berechnet werden:

$$L = \mathrm{cholesky}(\Sigma), \tag{3.14}$$

$$|\Sigma| = \prod_i L_{ii}, \tag{3.15}$$

$$\Sigma^{-1} = \left(L^{-1}\right)^T \cdot L^{-1}. \tag{3.16}$$

Da die Maximierung von $\ln\left(p\left(\mathbf{y}|\mathbf{x}\right)\right)$ unter Annahme konstanter $\mathbf{y}$ und $\mathbf{x}$ erfolgt, wird zur besseren Notation die folgende Umformulierung vorgenommen:

$$\ln\left(p\left(\mathbf{y}|\mathbf{x}\right)\right) \rightarrow \ln\left(p_{\mathbf{y}|\mathbf{x}}\left(\Theta\right)\right). \tag{3.17}$$

Für die Maximierung werden die Hyperparameter $\Theta$ gesucht, bei denen die erste Ableitung der Log-Marginal-Likelihood Null ergibt:

$$\frac{\mathrm{d}}{\mathrm{d}\Theta}\ln\left(p_{\mathbf{y}|\mathbf{x}}\left(\Theta\right)\right) \stackrel{!}{=} 0. \tag{3.18}$$

Wegen des Logarithmus folgt hieraus ohne weitere Rechnung:

$$\frac{\mathrm{d}}{\mathrm{d}\Theta}\ln\left(p_{\mathbf{y}|\mathbf{x}}\left(\Theta\right)\right) = \frac{1}{p_{\mathbf{y}|\mathbf{x}}\left(\Theta\right)}. \tag{3.19}$$

Die Nullstellensuche erfolgt mit dem Newton-Raphson-Verfahren, einem iterativen Ansatz zur Nullstellensuche. Dieses Verfahren ist für stetig-differenzierbare Funktionen anwendbar. Durch die gewählten Wahrscheinlichkeitsdichte- und Kovarianzfunktionen ist dieses Kriterium erfüllt. Durch die Maximierung werden die Hyperparameterwerte $\Theta_{\max}$ gefunden, bei denen die Log-Marginal-Likelihood maximal wird, die Daten also am Besten modelliert werden können.

Durch das Einsetzen von $\Theta_{\max} = (\zeta_{\max}, \xi_{\max})$ in Gleichung 3.7 lassen sich die Posterior-Kovarianzmatrizen für später noch genauer spezifizierte Zufallsvariablen $\mathbf{a}$ und $\mathbf{b}$ berechnen. Sie werden mit $K_{\text{post}}(\mathbf{a},\mathbf{b})$ (rauschfrei) und $\Sigma_{\text{post}}(\mathbf{a},\mathbf{b}) = K_{\text{post}}(\mathbf{a},\mathbf{b}) + \sigma_n^2\mathcal{I}$ (rauschbehaftet) bezeichnet.

Basierend auf dem so auf die bekannten Konturpunkte optimierten Posterior soll nun die Erstellung einer Prädiktion hergeleitet werden. Gauß-Prozesse gehen mit der Erfüllung der sogenannten Konsistenz-Eigenschaft einher [100]: Wenn ein Gauß-Prozess beispielsweise $(y_1, y_2) \sim \mathcal{N}(\mu, \Sigma)$ definiert, dann definiert er automatisch auch für eine Teilmenge der Variablen $y_1 \sim \mathcal{N}(\mu_1, \Sigma_{11})$, wobei $\Sigma_{11}$ die entsprechende Teilmatrix von $\Sigma$ ist. Für die Konturmodellierung entsprechen die Beobachtungen für die Trainingspunkte $y_1$ und die Beobachtungen für die zu prädizierenden Punkte $y_2$.

Mit dieser Erkenntnis lässt sich die Prädiktion für den Fall rauschfreier Beobachtungen als multivariate Gauß-Verteilung betrachten. Hierbei wird für den aus den Beobachtungen für die Trainingspunkte sowie die zu prädizierenden Beobachtungen der folgende Zufallsvektor formuliert:

$$\begin{bmatrix}\mathbf{f}\\ \mathbf{f}_*\end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix}K_{\text{post}}(\mathbf{x},\mathbf{x}) & K_{\text{post}}(\mathbf{x},\mathbf{x}_*)\\ K_{\text{post}}(\mathbf{x}_*,\mathbf{x}) & K_{\text{post}}(\mathbf{x}_*,\mathbf{x}_*)\end{bmatrix}\right), \tag{3.20}$$

mit

$\mathbf{x} \in \mathbb{R}^n$: $n$ Trainingspunkte

$\mathbf{x}_* \in \mathbb{R}^{n_*}$: $n_*$ zu prädizierende Punkte

$\mathbf{f} \in \mathbb{R}^n$: $n$ Beobachtungen für die Trainingspunkte

$\mathbf{f}_* \in \mathbb{R}^{n_*}$: $n_*$ Beobachtungen für die zu prädizierenden Punkte

Kovarianzmatrizen, die aus den Kombinationen von $\mathbf{x}$ und $\mathbf{x}_*$ entstehen:
$K_{\text{post}}(\mathbf{x}, \mathbf{x}) \in \mathbb{R}^{n \times n}$,
$K_{\text{post}}(\mathbf{x}, \mathbf{x}_*) \in \mathbb{R}^{n \times n_*}$,
$K_{\text{post}}(\mathbf{x}_*, \mathbf{x}) \in \mathbb{R}^{n_* \times n}$,
$K_{\text{post}}(\mathbf{x}_*, \mathbf{x}_*) \in \mathbb{R}^{n_* \times n_*}$: .

In Gleichung 3.20 ist mit $\mathbf{f}$ bereits ein Teil des Zufallsvektors bekannt. Der noch unbekannte Teil des Zufallsvektors, also $\mathbf{f}_*$, lässt sich berechnen, indem die bedingte Verteilung von $\mathbf{f}_*$ gegeben $\mathbf{x}, \mathbf{x}_*, f$ berechnet[2] wird:

$$\begin{aligned} \mathbf{f}_* | \mathbf{x}, \mathbf{x}_*, \mathbf{f} \sim \mathcal{N} \Big( & K_{\text{post}}(\mathbf{x}_*, \mathbf{x}) K_{\text{post}}(\mathbf{x}, \mathbf{x})^{-1} \mathbf{f}, \\ & K_{\text{post}}(\mathbf{x}_*, \mathbf{x}_*) - \\ & K_{\text{post}}(\mathbf{x}_*, \mathbf{x}) K_{\text{post}}(\mathbf{x}, \mathbf{x})^{-1} K_{\text{post}}(\mathbf{x}, \mathbf{x}_*) \Big). \end{aligned} \tag{3.21}$$

Der rauschbehaftete Fall lässt sich analog hierzu herleiten. Die nennenswerten Änderungen im Vergleich zu den Gleichungen 3.20 und 3.21 sind, dass zum einen die tatsächlichen $y$-Werte der Konturpunkte anstatt der durch den Gauß-Prozess beobachteten Punkte $f$ verwendet werden[3] und zum anderen im linken oberen Teil der Kovarianzmatrix für die Einträge auf der Hauptdiagonalen unkorreliertes Rauschen addiert wird.

Es ergeben sich:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N} \left( 0, \begin{bmatrix} K_{\text{post}}(\mathbf{x}, \mathbf{x}) + \sigma_n^2 \mathcal{I} & K_{\text{post}}(\mathbf{x}, \mathbf{x}_*) \\ K_{\text{post}}(\mathbf{x}_*, \mathbf{x}) & K_{\text{post}}(\mathbf{x}_*, \mathbf{x}_*) \end{bmatrix} \right) \tag{3.22}$$

2 Aus [100] von Rasmussen und von Mises in [83] folgend:
Es sei:

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \sim \mathcal{N} \left( \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} A & C \\ C^T & B) \end{bmatrix} \right).$$

Dann sind die bedingten Wahrscheinlichkeiten von $x$ gegeben $y$ und umgekehrt:

$$\mathbf{x} | \mathbf{y} \sim \mathcal{N} \left( \mu_x + C B^{-1} \left( \mathbf{y} - \mu_y \right), A - C B^{-1} C^T \right),$$
$$\mathbf{y} | \mathbf{x} \sim \mathcal{N} \left( \mu_y + C^T A^{-1} \left( \mathbf{x} - \mu_x \right), B - C^T A^{-1} C \right).$$

3 Siehe hierzu auch den Abschnitt 3.2 ab Seite 56.

und

$$\begin{aligned}\mathbf{f}_*|\mathbf{x},\mathbf{x}_*,\mathbf{y} \sim \mathcal{N}\Big( & K_{\text{post}}(\mathbf{x}_*,\mathbf{x})\Big[K_{\text{post}}(\mathbf{x},\mathbf{x})+\sigma_n^2\mathcal{I}\Big]^{-1}\mathbf{y}, \\ & K_{\text{post}}(\mathbf{x}_*,\mathbf{x}_*)- \\ & K_{\text{post}}(\mathbf{x}_*,\mathbf{x})\Big[K_{\text{post}}(\mathbf{x},\mathbf{x})+\sigma_n^2\mathcal{I}\Big]^{-1}K_{\text{post}}(\mathbf{x},\mathbf{x}_*)\Big).\end{aligned} \tag{3.23}$$

Für den Fall, das nur ein einzelner Konturpunkt gleichzeitig prädiziert wird[4], sowie mit der verkürzten Schreibweise $\Sigma_{\text{post}} = K_{\text{post}}(\mathbf{x},\mathbf{x}) + \sigma_n^2\mathcal{I}$, ergibt sich für die prädizierende Beobachtung $f_*$ an der Stelle $x_p$:

$$\begin{aligned}f_* \sim \mathcal{N}\Big( & \mathbf{k}_{\text{post}}(x_p,\mathbf{x})\Sigma_{\text{post}}^{-1}\mathbf{y}, \\ & k_{\text{post}}(x_p,x_p) - \mathbf{k}_{\text{post}}(x_p,\mathbf{x})\Sigma_{\text{post}}^{-1}\mathbf{k}_{\text{post}}(\mathbf{x},x_p)\Big),\end{aligned} \tag{3.24}$$

mit den aus den detektierten Konturpunkten gewonnenen Trainingsdaten $\mathbf{x}$, $\mathbf{y}$ sowie $x_p$ als $x$-Koordinate für die Vorhersage $f_*$.

In Gleichung 3.24, wie auch in den Gleichungen 3.21 und 3.23, ist bemerkenswert, dass zum einen der Mittelwert der Gauß-Verteilung eine lineare Funktion der y-Werte der detektierten Kontur ist und zum anderen die Unsicherheit der Prädiktion nur von den Kovarianzen zwischen allen x-Werten (Trainingspunkte und zu prädizierende Punkte) abhängt und unabhängig von den prädizierten y-Werten ist.

## 3.6 KONTUREXTRAPOLATION

Um nun eine mit einem Posterior modellierte Kontur zu extrapolieren, werden Prädiktionen $y_p$ an den Stellen $x_p$ mittels des auf den bekannten Konturpixeln optimierten Gauß-Prozesses erzeugt. $x_p$ sind hierbei die $x$-Koordinaten für die zu extrapolierende Kontur. Deshalb werden alle $x$-Werte des aktuell codierten Bereichs im lokalen Koordinatensystem, also $s \leq x < 2s$ bei einer Blockgröße $s$, eingesetzt. Die $y$-Koordinaten werden jeweils durch Auswertung des Posterior-Erwartungswerts ermittelt:

$$y_p = m_{\text{post}}(x_p) = \mathbf{k}_{\text{post}}(x_p,\mathbf{x})\Sigma^{-1}\mathbf{y}. \tag{3.25}$$

Die prädizierten Konturpunkte sind dann $\mathbf{p} = (\mathbf{x}_p, \mathbf{y}_p)$. Es erfolgt eine Rundung auf Ganz-Pel-Genauigkeit.

4 Dieser Fall entspricht der für Kapitel 5 genutzten Implementierung des vorgeschlagenen Verfahrens.

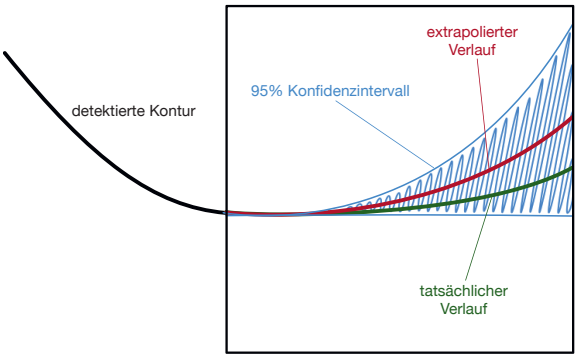



Abbildung 3.4: Konturextrapolation mit 95%-Konfidenzintervallen ($y_p \pm 1.96 \cdot \sigma_p$)

An dieser Stelle ließe sich ermitteln, ob die prädizierten Konturpunkte innerhalb des aktuell codierten Blockes liegen, also

$$s \leq y_p < 2s \tag{3.26}$$

genügen. Aus Gründen einer geordneten Software-Architektur ist es jedoch erstrebenswert, dass das vorgeschlagene Konturmodell nicht nur mit dem in Kapitel 4 vorgeschlagenen Verfahren zur Abtastwertprädiktion mit neuronalen Netzwerken harmoniert, sondern auch vollständig kompatibel zu dem in der eigenen Vorarbeit [69] vorgeschlagenen *Along-contour*-Verfahren zur Abtastwertprädiktion ist. Da in letzterem die extrapolierte Kontur entlang der durch sie geschnittenen Blockgrenze verschoben wird, können eigentlich außerhalb des aktuellen Blockes liegende Konturpixel in diesen hineinverschoben werden. Folglich wird die genannte Prüfung im nachfolgenden Kapitel 4 im Rahmen der Erzeugung der Eingangsdaten für das neuronale Netzwerk diskutiert werden.

Die Unsicherheit der Prädiktion lässt sich über die Varianz beziehungsweise Standardabweichung berechnen:

$$\sigma_p^2 = k_{\text{posterior}}(x_p, x_p), \tag{3.27}$$

$$\sigma_p = \sqrt{\sigma_p^2}. \tag{3.28}$$

Ein Beispiel für die Konturextrapolation ist in Abbildung 3.4 visualisiert. Hierbei ist neben dem extrapolierten und dem tatsächlichen Konturverlauf die Konfidenz in der Extrapolation mit 95%-Konfidenzintervallen ($y_p \pm 1.96 \cdot \sigma_p$) dargestellt.

Ebenfalls im nachfolgenden Kapitel wird beschrieben werden, wie aus der Varianz ein zusätzlicher Kanal für die Eingangsdaten des neuronalen Netzwerks erzeugt wird.

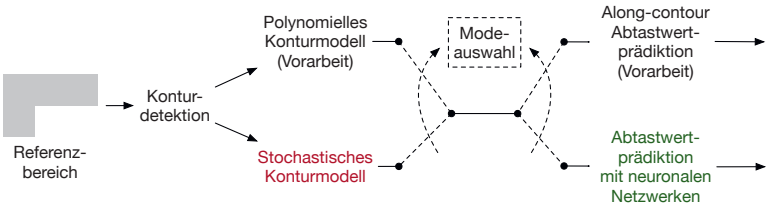



Abbildung 3.5: Einbettung der entwickelten Verfahren in das Gesamtsystem: Die im Rahmen dieser Arbeit entwickelten Verfahren zur Konturmodellierung und Abtastwertprädiktion werden so entworfen, dass sie unter Wahrung der vollständigen Kompatibilität mit dem jeweils entsprechenden Verfahren aus den Vorarbeiten getauscht werden können. Hierdurch kann der Mehrwert durch die in dieser Arbeit vorgeschlagenen Verfahren ermittelt werden.

## 3.7 EINBETTUNG IN DAS GESAMTSYSTEM

Als Abschluss dieses Kapitels wird erläutert, wie sich das vorgeschlagene Verfahren zur Modellierung von Konturen in das Gesamtsystem *CoMIC* einordnet. Die Einordnung wird anhand von Abbildung 3.5 erläutert. Das Gesamtsystem umfasst eine Verkettung von mehreren Verfahren. Die Konturdetektion steht in jedem Fall zu Beginn der Verarbeitung. Der Rest der Verfahrenskette ist zweigeteilt in die Konturmodellierung und die Abtastwertprädiktion. Diese Teilung folgt der bereits in Kapitel 1 näher begründeten Prämisse, dass die Konturmodellierung geeignet ist, um die globalen Eigenschaften des Signals, beispielsweise Objektformen, gut zu erfassen, während die auf lokalen Signaleigenschaften aufbauende Abtastwertprädiktion gut für die Verarbeitung von lokalen Signaleigenschaften wie Farbverläufen ist.

Für jeden dieser beiden Blöcke wird im Rahmen dieser Arbeit ein neues Verfahren vorgeschlagen. Hierbei ist das vorgeschlagene Verfahren jeweils eine Alternative zu dem entsprechenden Verfahren aus den eigenen Vorarbeiten [65, 69]. Das in diesem Kapitel vorgeschlagene stochastische Konturmodell bietet eine Alternative zu dem polynomiellen Konturmodell. Das im nächsten Kapitel vorgeschlagene Verfahren zur Abtastwertprädiktion mit neuronalen Netzwerken lässt sich als Alternative zum *Along-contour*-Verfahren einsetzen.

Die System-Architektur — und damit einhergehend auch die das System implementierende Software-Architektur — ist so entworfen, dass für beide Teile der Verfahrenskette das Verfahren nach Belieben durch die beiden Schalter in Abbildung 3.5 gewählt werden kann und unabhängig von der Wahl die Module kompatibel zueinander sind. Die Ausgabe

der Konturmodellierung, und damit die an der Schnittstelle zwischen Konturmodellierung und Abtastwertprädiktion weitergegebenen Informationen, sind pro modellierter Kontur der Schnittpunkt der Kontur mit der Blockgrenze des aktuell codierten Blocks, der Verlauf der extrapolierten Kontur, die Zuordnung von unabhängiger und abhängiger Variable zu $x$ und $y$ sowie die Konfidenz der Extrapolation. Letztere steht nur für das stochastische Konturmodell zur Verfügung.

# 4

# VERFAHREN ZUR ABTASTWERTPRÄDIKTION MITTELS MASCHINELLEN LERNENS

In diesem Kapitel wird das vorgeschlagene Verfahren zur Abtastwerprädiktion mittels maschinellen Lernens beschrieben. Nach einer kurzen Einordnung in das Gesamtsystem wird die für das Training der neuronalen Netzwerke verwendete Datenbasis erläutert. In den zentralen Abschnitten des Kapitels werden die verwendeten Architekturen sowie das Trainingsverfahren diskutiert.

## 4.1 EINORDNUNG IN DAS GESAMTSYSTEM

Die Eingangssignale für das in diesem Kapitel beschriebene Verfahren sind die Abtastwerte des Referenzbereichs und pro Kontur die Ausgaben der Konturextrapolation, also der Schnittpunkt der Kontur mit dem zu codierenden Block, die Zuordnung unabhängiger und abhängiger Variablen, der Verlauf der extrapolierten Kontur sowie die zugehörige Konfidenz in die Extrapolation. In Abhängigkeit des gewählten Konturmodellierungsverfahrens (vgl. Abbildung 3.5) ist das letztgenannte Eingangssignal für polynomiell modellierte Konturen nicht verfügbar. Das Ergebnis des vorgeschlagenen Verfahrens sind die prädizierten Abtastwerte für den zu codierenden Block.

## 4.2 DATENBASIS

Die richtige Wahl von Trainingsdaten ist von entscheidender Bedeutung für den Erfolg des Trainings neuronaler Netzwerke. Es ist erforderlich, dass die Anzahl an Beispielen in den Trainingsdaten hoch genug ist, da Netzwerke in der Regel viele zu trainierende Parameter in ihrer Konfiguration haben. Des Weiteren müssen die Trainingsdaten vielfältig genug sein, um eine gute Generalisierung zu ermöglichen. Auch müssen die Daten frei von Artefakten sein, die das Training behindern können und zu dem Erlernen von irrelevanten Merkmalen führen.

Im Rahmen dieser Arbeit wird die Raw Image Dataset (RAISE)-Datenbank [19] verwendet. Diese Datenbank beinhaltet Fotografien, die mit verschiedenen Kameras aufgenommen wurden, und Auflösungen von 3008 ×

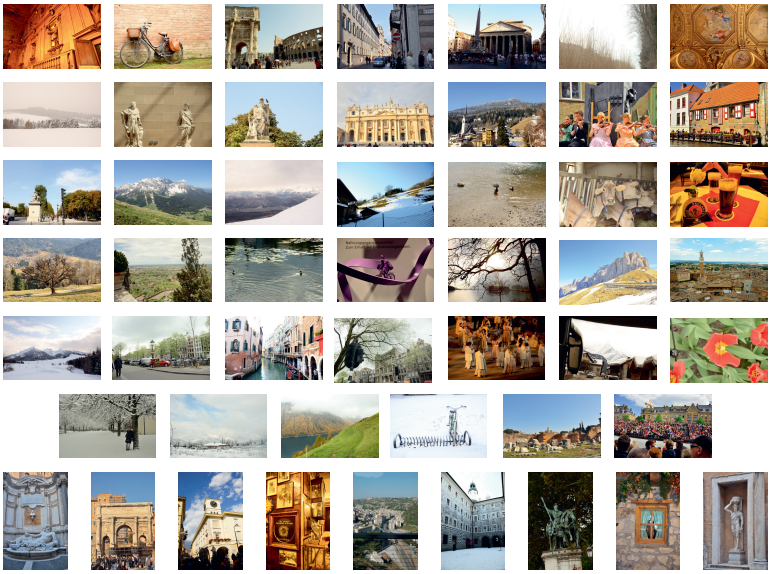


Abbildung 4.1: Bilder der Trainingsdatenbank.

2000, $4288 \times 2848$ und $4928 \times 3264$ haben. Die Bilder wurden als *Raw*-Aufnahmen von den Kameras gespeichert und auch im weiteren Verlauf der Datenbankerstellung nie verlustbehaftet codiert. Die Verwendung von solchen unkomprimierten[1] Bildern als Trainingsdaten ist erstrebenswert, weil die Bilder keinerlei Codierungsartefakte enthalten. Hierdurch wird verhindert, dass die Netzwerke während des Trainings lernen, Merkmale aus Codierungsartefakten zu extrahieren. Die Motive der Bilder in der Datenbank sind sehr vielfältig. Von der Autoren werden diese in die Kategorien *Outdoor*, *Indoor*, *Landscape*, *Nature*, *People*, *Objects* und *Buildings* klassifiziert [19]. Zur Erzeugung der Trainingsdaten werden 50 Bilder zufällig ausgewählt. Die verwendeten Bilder aus der Datenbank sind in Abbildung 4.1 dargestellt.

Aus den Bildern der RAISE-Datenbank werden bis zu vier verschiedene Eingangssignale für die neuronalen Netzwerke gewonnen. Diese werden im Folgenden erläutert:

1 Es ist anzumerken, dass der Referenzbereich während der Codierung durch die Quantisierung mit unterschiedlichen Quantisiererstufenbreiten nicht als unkomprimiert angesehen werden kann. Im Rahmen dieser Arbeit wird nicht weiter untersucht, inwiefern sich die Effizienz des vorgeschlagenen Verfahrens durch die Verwendung von quantisierten Trainingsdaten oder durch das Training unterschiedlicher Netzwerke für verschiedene Quantisierungsparameter weiter steigern ließe.

1. Abtastwerte des Referenzbereichs: Die Referenzabtastwerte sind die wichtigste Information für das neuronale Netzwerk, da dieses die Aufgabe hat, die Abtastwerte des benachbarten, aktuell zu codierenden, Blockes zu prädizieren. Anders als bei anderen Problemen, beispielsweise im Feld der Klassifizierung, in dem mehr die relativen Änderungen zwischen benachbarten Abtastwerten relevant sind, sind bei dem Problem der Abtastwertprädiktion die absoluten Werte von Interesse. Deshalb erfolgt lediglich eine Skalierung des ursprünglichen Wertebereichs von 0 bis 255 auf den Wertebereich 0,0 bis 1,0, aber keine weitere Manipulation.

2. Verfügbarkeit der Referenzwerte als Maske: Je nach gewählter Partitionierung und der Position im Bild mit Bezug auf die Bild- und Slice-Grenzen sind gegebenenfalls nicht alle Referenzabtastwerte verfügbar. Das Wissen hierüber ist von Relevanz für das Netzwerk, da natürlich nur existierende Abtastwerte in eine Prädiktion eingehen sollten. Gleichwohl ist es unpraktikabel, die Größe und die Form der Eingangstensoren an die verfügbaren Referenzabtastwerte anzupassen. Um dem neuronalen Netzwerk dennoch die Kenntnis über die Verfügbarkeit des Referenzbereichs anzuzeigen wird eine binäre Maske, dessen örtliche Größe mit der Größe des Referenzbereichs übereinstimmt, verwendet. Es könnte eingewendet werden, dass das Wissen über das Nichtvorhandensein der Abtastwerte auch aus denselben extrahiert werden könne, beispielsweise in dem diese auf Null gesetzt werden. Dieses ist jedoch nicht der Fall, da es sich hierbei um einen validen Eingangswert handelt, der in realen Signalen nicht zuletzt bei Unterbelichtung auftreten kann.

3. Modellierte und extrapolierte Kontur: Der Verlauf von Konturen, und damit die Kenntnis über die Form von Objekten innerhalb des betrachteten Bildausschnitts, wird als wertvolle Information für die Abtastwertprädiktion angesehen. Aus der Vorarbeit [24] ist bekannt, dass die alleinige Verwendung von Referenzabtastwerten, unter der Annahme, dass das neuronale Netzwerk die Schätzung des Konturverlaufs mitlernen könne, nicht zu zufriedenstellenden Ergebnissen führt. Der Konturverlauf wird in Form eines ebenfalls binären Konturbildes in das Netzwerk eingegeben. Das Signal hat die gleiche örtliche Auflösung wie der zu einem Rechteck erweiterte Referenzbereich. Hierfür wird die gefundene Kontur, die auch schon die Eingabe der Konturmodellierung war, mit dem Ergebnis der Konturextrapolation kombiniert.

4. Konfidenz der Konturextrapolation: Mit dem Ergebnis der stochastischen Konturmodellierung und der anschließenden Konturextrapolation hat man für jede extrapolierte Stelle $x_p$ entsprechend Gleichung 3.27 eine Konfidenz $\sigma_p^2$. Aus dieser wird, wie nachfolgend noch weiter ausgeführt, ein Grauwert-Signal mit der gleichen örtlichen Größe wie bei den anderen Eingangssignalen erzeugt.

Durch die Verarbeitung der 50 zufällig gewählten Bilder aus der RAISE-Datenbank werden Hierarchical Data Format 5 (HDF5)-Datenbanken [123] mit den Trainings- und Validierungsdaten erzeugt. Es werden getrennte Datenbanken für die unterschiedlichen Blockgrößen erstellt. Dieses Datenbank-Format wurde gewählt, weil es eine hohe Effizienz beim wahlfreien Lesen von Teilabschnitten der Datenbank ermöglicht. Dieses ist die typische Anforderung während des Trainingsvorgang, in dem die Daten in einer zufälligen Reihenfolge gelesen werden. Alle Werte werden mit 32-Bit Genauigkeit als Fließkommazahlen gespeichert, da diese für die Ein- und Ausgänge des neuronalen Netzwerks benötigt werden. Hierdurch ist zwar der Speicherplatzbedarf höher als bei einer Speicherung von 8-Bit Daten, jedoch ist die einmalige Konvertierung praktikabler als eine wiederkehrende Konvertierung in jeder Trainingsiteration.

Für die Datenbankerzeugung wird über alle für die Datenerzeugung vorgesehenen Bilder iteriert. Jedes Bild wird entsprechend der gewählten Blockgröße in nicht überlappende Blöcke aufgeteilt, über welche ebenfalls iteriert wird. Die Abtastwerte in direkt nebeneinander liegenden Blöcken sind insbesondere für kleine Blockgrößen stark korreliert. Zur Vermeidung unnötiger Redundanzen in der Trainingsdatenbank wird in Abhängigkeit der Blockgröße nur jeder $n$-te Block verwendet, wobei $n$ die folgenden Werte hat: 1 für $64 \times 64$-Blöcke, 2 für $32 \times 32$-Blöcke, 6 für $16 \times 16$-Blöcke, 25 für $8 \times 8$-Blöcke und 100 für $4 \times 4$-Blöcke. Für jeden Block erfolgt die Konturdetektion wie im vorherigen Kapitel beschrieben. Es kann eine Fallunterscheidung getroffen werden: Falls keine Konturen im zugehörigen Referenzbereich des Blockes sind, werden für diesen Datenbankeintrag lediglich die Felder für die Referenzabtastwerte sowie die Verfügbarkeitsmaske der Referenzabtastwerte befüllt.

Im Normalfall gibt es Konturen. Die werden mit dem vorgeschlagenen Konturmodell aus Kapitel 3 verarbeitet. Um eine robuste Konturmodellierung sicherzustellen, werden — wie auch bei der eigentlichen Codierung — bei der Trainingsdatenerzeugung nur Konturen mit mehr als drei Konturpixeln extrapoliert. Kürzere Konturen werden nicht betrachtet. Alle geeigneten Konturen werden modelliert und extrapoliert. Das Ergebnis der Konturextrapolation besteht aus den Vorhersagen gemäß Gleichung 3.25 mit den zugehörigen Konfidenzen gemäß Gleichung 3.27. Hierbei handelt sich pro Konturpunkt um insgesamt drei Zahlenwerte:

Zwei Zahlenwerte für die Koordinaten des Punkts, $\left(x_{pi}, y_{pi}\right)$, und einen Zahlenwert für die Varianz, $\sigma_{pi}^2$.

Die für das vorgeschlagene Verfahren verwendeten Netzwerkarchitekturen sind jedoch wegen ihrer zugrunde liegenden Faltungsschichten für die Verarbeitung von zweidimensionalen Daten ausgelegt. Deshalb erfolgt eine Umwandlung in ein entsprechendes Format. Dass die Umwandlung von eigentlich nicht zweidimensionalen Daten in solche für eine Verarbeitung durch Faltungsnetzwerke sinnvoll ist, ist beispielsweise aus dem Fachgebiet der Audioverarbeitung bekannt [142]. In der genannten Arbeit werden Audiodaten durch die Umwandlung in eine Spektrumsdarstellung als zweidimensionale Bilder interpretiert.

Das binäre Konturbild ergibt sich aus den Koordinaten der Konturpunkte. Aus der Konfidenz in die Konturextrapolation wird eine sogenannte Konfidenzkarte erzeugt. Die Erläuterung des Vorgehens erfolgt anhand von Abbildung 4.2. Für die Erzeugung der Konfidenzkarte wird $\sigma_p^2$ aus Gleichung 3.27 für jedes $x_p$ im zu codierenden Block berechnet. Zur vereinfachten Notation wird ein lokales Koordinatensystem, dessen Ursprung im extrapolierten Konturpunkt $\left(x_p, y_{x_p}\left(x_p\right)\right)$ liegt, angenommen. In jeden Konturpunkt wird eine Gauß-förmige Konfidenzfunktion $c_{x_p}\left(y_{x_p}\right)$ mit der berechneten Varianz gelegt:

$$c_{x_p}\left(y_{x_p}\right) = \frac{1}{\sqrt{2\pi\sigma_p^2}}\, e^{-\frac{y_{x_p}^2}{2\sigma_p^2}} \tag{4.1}$$

Die Auswertungen der jeweiligen Konfidenzfunktionen in jedem Punkt entlang der $y_{x_p}$-Achsen werden als Grauwerte interpretiert. Das Beispiel in Abbildung 4.3 zeigt exemplarisch eine auf die geschilderte Weise entstehende Konfidenzkarte.

Im Gegensatz zu anderen Arbeiten wie [66] werden nicht die durch eine Codierung entstehenden rekonstruierten Abtastwerte für die Trainingsdaten verwendet sondern die originalen Abtastwerte. Zum einen wäre bei der Verwendung vor einer Codierung zu entscheiden, mit welchen Quantisierungsparametern die Daten zu codieren wären. Eine Optimierung der Trainingsdaten — oder sogar noch weitergehend das Trainieren von spezifischen Netzwerken — auf die in akademischen Veröffentlichungen und Standardisierungsbeiträgen typischerweise verwendeten vier Quantisierungsparameter erscheint nicht sinnvoll, da in realen Szenarien weit mehr unterschiedliche Quantisierungsparameter verwendet werden. Die Unterschiede zwischen den rekonstruierten Abtastwerten und den Original-Abtastwerten liegen im Wesentlichen in der durch die Transformationscodierung bedingte leichte Tiefpassfilterung sowie in

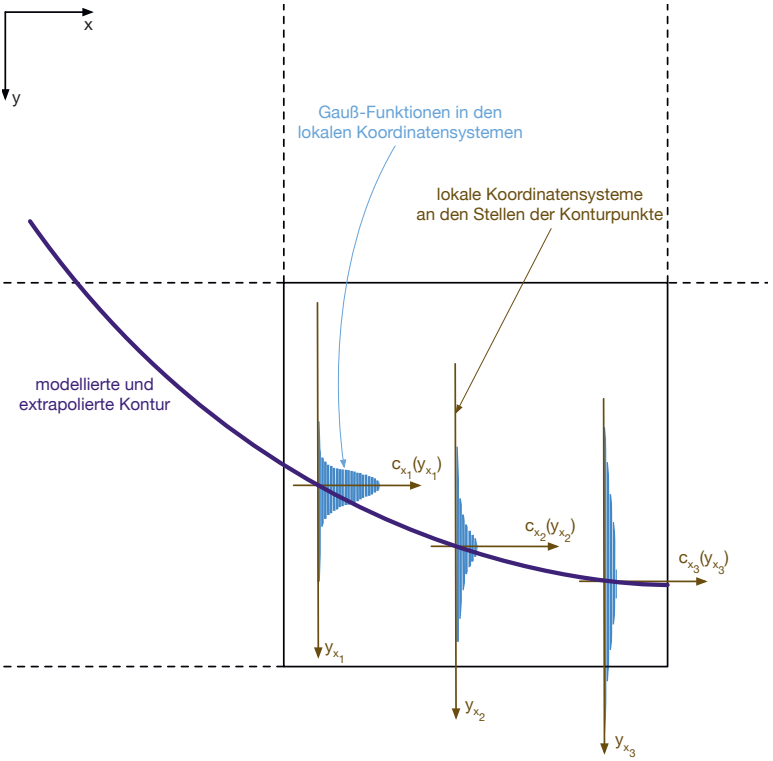



Abbildung 4.2: Erzeugung der Konfidenzkarte: In jeden Punkt der extrapolierten Kontur wird eine Gauß-Funktion entsprechend Gleichung 4.1 gelegt, deren Varianz durch die Konfidenz der Extrapolation gemäß Gleichung 3.27 gegeben ist.

Codierungsartefakten begründet. Da keine Codierungsartefakte in den Trainingsdaten vorhanden sind, wird das neuronale Netzwerk nicht die Reproduktion von solchen Artefakten erlernen. Durch eine Tiefpassfilterung würden hochfrequente Daten in den Trainingsdaten fehlen. Es wird jedoch angenommen, dass neuronale Netzwerke hohe Frequenzen in Bildsignalen, bis hin zu dem Rauschanteil, nicht pixelgenau vorhersagen können. Ein künstlich erzeugtes nicht pixel-genaues Rauschen würde jedoch zu einer ansteigenden Datenrate für den Prädiktionsfehler führen. Ferner geht der Verzicht auf die hohen Signalanteile einher mit der später beschriebenen Wahl der Kostenfunktion für das Training der Netzwerke. Durch diese Kostenfunktion erfolgt eine Regularisierung gegen höhere Frequenzen im prädizierten Signal. Hierdurch macht es letztendlich

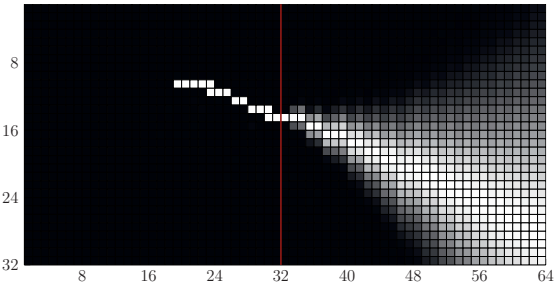



Abbildung 4.3: Beispiel einer Konfidenzkarte

keinen Unterschied, ob hohe Frequenzen in den Trainingsdaten sind oder nicht, da diese durch die Kostenfunktion zwar nicht verboten aber bestraft würden.

Die Eigenschaften des Luma- und der Chroma-Signale unterscheiden sich voneinander. Das Luma-Signal hat eine wesentlich höhere Entropie als die Chroma-Signale. Deshalb ist es nicht sinnvoll, Luma- und Chroma-Signale einheitlich zu betrachtet. Stattdessen werden getrennte Datenbanken erzeugt und auch separate neuronale Netzwerke trainiert.

## 4.3 ARCHITEKTUREN

Autoencoder sind eine Ausführungsform von neuronalen Netzwerken. Der Grundgedanke eines klassischen Autoencoders[2] besteht aus einer Architektur mit der das Ziel verfolgt wird, dass die Eingangsdaten den Ausgangsdaten entsprechen. In der sogenannten Codierungsschicht, die auch als *Bottleneck* oder *Latent Space* bezeichnet wird, in der Mitte der Architektur werden die Daten mit einer deutlich geringeren Anzahl an Neuronen repräsentiert als am Ein- beziehungsweise Ausgang. Der Teil der Architektur vor der Codierungsschicht wird als Encoder bezeichnet, der Teil danach als Decoder[3]. Ein exemplarischer Autoencoder ist in Abbildung 4.5 abgebildet. Im Encoder-Teil des Autoencoders werden die Daten von Schicht zu Schicht mit einer geringeren Kapazität repräsentiert.

2 Der Begriff *Autoencoder* wird verwendet, um die Konsistenz mit der Literatur zur wahren. Da die so bezeichneten Netzwerke sowohl einen Encoder- als auch einen Decoder-Teil enthalten, wäre aus Sicht der Codierungsnomenklatur der Begriff *Autocodec* treffender.

3 Die beiden Hälften des Autoencoders werden als Encoder und Decoder bezeichnet. Diese Bezeichnung ist unabhängig von der Verwendung der vorgeschlagenen Verfahren im Encoder- und Decoder-Teil der betrachteten Bild- und Videocodecs. In beiden Hälften der Bild- und Videocodecs werden sowohl der Encoder- als auch der Decoder-Teil des Autoencoders verwendet.

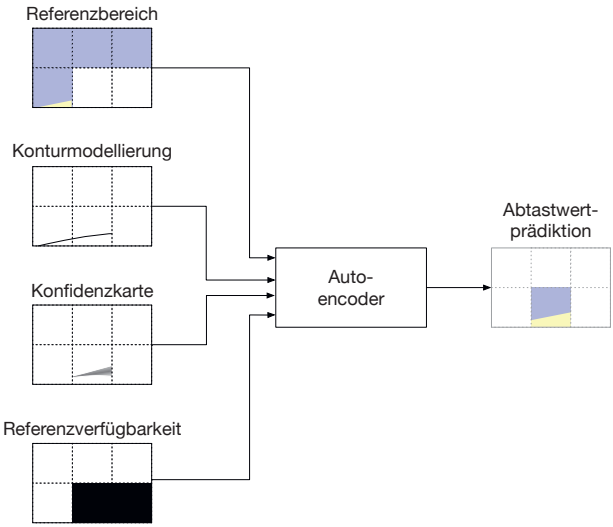



Abbildung 4.4: Ein- und Ausgabedaten für den Autoencoder. Der Autoencoder hat bis zu vier Eingangssignale: Immer vorhanden sind der Referenzbereich sowie die Referenzverfügbarkeit. Zusätzlich werden je nach trainiertem Modell die Konturmodellierung (inklusive Extrapolation) und/oder die Konfidenzkarte als weitere Eingänge verwendet.

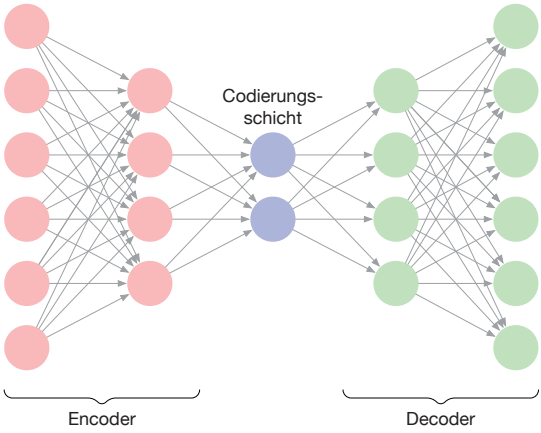



Abbildung 4.5: Exemplarische Veranschaulichung eines Autoencoders

Im Decoder-Teil wird die Kapazität mit jeder Schicht bis zur Zielgröße erhöht. Hierdurch werden die wesentlichen Eigenschaften des Signals erkannt und bleiben durch die entsprechende Repräsentation in der Codierungsschicht erhalten. Weniger relevante Eigenschaften im Signal bleiben nicht erhalten. Der ursprüngliche Anwendungsfall — hierher kommt auch der Name Autoencoder — ist die Codierung von Informationen. Durch die Eigenschaft, dass die wesentlichen Merkmale im Signal erhalten bleiben oder weiter verarbeitet werden, sind Autoencoder auch für Anwendungen wie die Entrauschung oder die nachträgliche Kolorierung von monochromen Bildern gut geeignet.

Das Ziel des in diesem Kapitel vorgeschlagenen Verfahrens ist ähnlich: Das neuronale Netzwerk soll die relevanten Signalcharakteristiken des Referenzbereichs identifizieren und basierend auf ihnen eine Prädiktion der Abtastwerte im zu codierenden Block erstellen. Ein wesentlicher Unterschied ist, dass in diesem Szenario nicht die gleichen Signale am Ein- und Ausgang vorhanden sind. Am Eingang liegen zwei bis vier Signale mit der Größe des auf ein Rechteck erweiterten Referenzbereichs an. Das Ausgangssignal hat die Größe des zu codierenden Blocks und beinhaltet die prädizierten Abtastwerte. Eine Darstellung der ein- und ausgegebenen Daten befindet sich in Abbildung 4.4. Die Kapazität der Codierungsschicht ist so zu wählen, dass genug relevante Informationen aus den bis zu vier Eingängen extrahiert werden können, um eine gute Prädiktion des zu codierenden Blocks zu erzielen. Wegen der konzeptionellen Eignung von Autoencodern werden diese für das vorgeschlagene Verfahren verwendet. Da Bilddaten verarbeitet werden, werden ferner Faltungsautoencoder, engl. Convolutional Auto Encoder (CAE), verwendet.

Nachfolgend wird zunächst die Architektur für zu codierende Blöcke der Größe $32 \times 32$ beschrieben. Der grundsätzliche Aufbau für die Netzwerke mit anderen Eingangsgrößen ist gleich. Einige Anpassungen von einzelnen Parametern sind beschrieben. Eine Übersicht über die Details der Konfigurationen der verwendeten Architekturen gibt es in Tabelle 4.1.

Im Prinzip wird im Encoder-Teil die Anzahl an Filtern mit jeder Schicht verdoppelt während gleichzeitig die örtliche Auflösung der durch das Netzwerk prozessierten Tensoren halbiert wird. Im Decoder findet der umgekehrte Prozess statt: Die Anzahl an Filtern halbiert sich mit jeder weiteren Schicht während die örtliche Auflösung verdoppelt wird. Jeweils in der ersten Schicht wird die rechteckige örtliche Auflösung der Eingangssignale durch asymmetrische Filtergrößen und eine asymmetrische Schrittweite in Tensoren mit quadratischer örtlicher Auflösung überführt. Alle weiteren Filter sind symmetrisch und werden ebenfalls mit symmetrischen Schrittweiten verschoben.

Tabelle 4.1: Architekturübersicht. Die Nomenklatur für die Dimensionen ist Kanalanzahl×Breite×Höhe. EF steht für die Eingangsfilteranzahl, BF für die Anzahl an Filtern in der Codierungsschicht (engl. *Bottleneck*). Der grundsätzliche Aufbau ist für alle Blockgrößen gleich: Im Encoder steigt mit jeder tiefergehenden Schicht die Anzahl an Filtern mit dem Faktor zwei während die örtliche Auflösung der Tensoren halbiert wird. Im Decoder ist es anders herum. Die Netzwerke für die kleineren Blockgrößen haben eine geringere Anzahl an Schichten.

| | **8×8** | | **16×16** | | **32×32** | | **64×64** | |
|---|---|---|---|---|---|---|---|---|
| | Aufbau | Dimension | Aufbau | Dimension | Aufbau | Dimension | Aufbau | Dimension |
| 1 | Eingabe | 3×24×16 | Eingabe | 3×48×32 | Eingabe | 3×96×64 | Eingabe | 3×192×128 |
| 2 | Conv2d<br>BatchNorm<br>Dropout<br>LeakyReLU | EF×24×16 | Conv2d<br>BatchNorm<br>Dropout<br>LeakyReLU | EF×48×32 | Conv2d<br>BatchNorm<br>Dropout<br>LeakyReLU | EF×32×32 | Conv2d<br>BatchNorm<br>Dropout<br>LeakyReLU | EF×192×128 |
| 3 | Conv2d<br>BatchNorm<br>Dropout<br>LeakyReLU | 2•EF×8×8 | Conv2d<br>BatchNorm<br>Dropout<br>LeakyReLU | 2•EF×16×16 | Conv2d<br>BatchNorm<br>Dropout<br>LeakyReLU | 2•EF×16×16 | Conv2d<br>BatchNorm<br>Dropout<br>LeakyReLU | 2•EF×64×64 |
| 4 | Conv2d<br>BatchNorm<br>Dropout<br>LeakyReLU | 4•EF×4×4 | Conv2d<br>BatchNorm<br>Dropout<br>LeakyReLU | 4•EF×8×8 | Conv2d<br>BatchNorm<br>Dropout<br>LeakyReLU | 4•EF×8×8 | Conv2d<br>BatchNorm<br>Dropout<br>LeakyReLU | 4•EF×32×32 |
| 5 | Conv2d<br>BatchNorm<br>Dropout<br>LeakyReLU | BF×4×4 | Conv2d<br>BatchNorm<br>Dropout<br>LeakyReLU | BF×4×4 | Conv2d<br>BatchNorm<br>Dropout<br>LeakyReLU | 8•EF×4×4 | Conv2d<br>BatchNorm<br>Dropout<br>LeakyReLU | 8•EF×16×16 |
| 6 | ConvTransp2d<br>BatchNorm<br>LeakyReLU | 4•EF×4×4 | ConvTransp2d<br>BatchNorm<br>LeakyReLU | 4•EF×8×8 | Conv2d<br>BatchNorm<br>Dropout<br>LeakyReLU | BF×2×2 | Conv2d<br>BatchNorm<br>Dropout<br>LeakyReLU | BF×8×8 |
| 7 | ConvTransp2d<br>BatchNorm<br>LeakyReLU | 2•EF×8×8 | ConvTransp2d<br>BatchNorm<br>LeakyReLU | 2•EF×16×16 | ConvTransp2d<br>BatchNorm<br>LeakyReLU | 8•EF×4×4 | ConvTransp2d<br>BatchNorm<br>LeakyReLU | 8•EF×16×16 |
| 8 | ConvTransp2d<br>BatchNorm<br>Tanh | 1×8×8 | ConvTransp2d<br>BatchNorm<br>Tanh | 1×16×16 | ConvTransp2d<br>BatchNorm<br>LeakyReLU | 4•EF×8×8 | ConvTransp2d<br>BatchNorm<br>LeakyReLU | 4•EF×32×32 |
| 9 | | | | | ConvTransp2d<br>BatchNorm<br>LeakyReLU | 2•EF×16×16 | ConvTransp2d<br>BatchNorm<br>LeakyReLU | 2•EF×32×32 |
| 10 | | | | | ConvTransp2d<br>BatchNorm<br>LeakyReLU | EF×32×32 | ConvTransp2d<br>BatchNorm<br>LeakyReLU | EF×64×64 |
| 11 | | | | | ConvTransp2d<br>BatchNorm<br>Tanh | 1×32×32 | ConvTransp2d<br>BatchNorm<br>Tanh | 1×64×64 |
| | 1,5M Parameter | | 4,5M Parameter | | 9,7M Parameter | | 5,5M Parameter | |

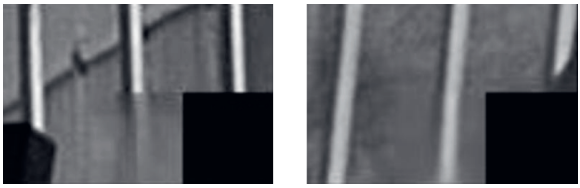


Abbildung 4.6: Vergleich des Einflusses der Filtergröße. Wird die Filtergröße zu klein gewählt, dann können Strukturen nicht gut prädiziert werden (links). Bei der für die vorgeschlagenenen Netzwerke gewählten Filtergröße können Strukturen sinnvoll extrapoliert werden (rechts).

Bei der Wahl der Filtergröße sind mehrere Abwägungen zu treffen. Einerseits ist eine gewisse Mindestgröße erforderlich, um die für die Abtastwertprädiktion notwendige Zusammenhänge im Signal erkennen zu können. Andererseits führen größere Filter auch zu einer höheren Verringerung der örtlichen Auflösung an den Rändern der verarbeiteten Tensoren[4]. Dieses führt zu dem Problem, dass die umsetzbarer Tiefe des Netzwerkes ebenfalls sinkt. Wird die Tiefe zu gering, dann reicht die Kapazität des Netzwerks nicht mehr für eine gute Prädiktion der Abtastwerte. Aufgrund von entsprechenden Analysen in Vorarbeiten wird die Filtergröße zu $4 \times 4$ und die Schrittweite zu $2 \times 2$ gewählt. Kleinere Filter ermöglichen keine Fortsetzung von strukturierten Signalanteilen, wie im Beispiel der Abbildung 4.6 dargestellt ist.

Würden die Netzwerke so wie bis hierhin beschrieben verwendet werden, dann käme es beim Training zu einer Überanpassung auf die Trainingsdaten. Batch-Normierung ist ein anerkanntes Verfahren, um die Lernfähigkeit neuronaler Netzwerke zu verbessern. Die ursprüngliche Motivation liegt in der Beobachtung, dass während des Trainings bereits jeweils kleine Änderungen der Gewichte in den Schichten eines tiefen Netzwerkes durch die multiplikative Verknüpfung der Schichten zu großen Änderungen der Aktivierungen führen [50]. Hierdurch müssten diese Änderungen der Aktivierungen in jeder Trainingsiteration kompensiert werden. Durch die Normierung der Eingänge für jedes Batch entfällt diese Notwendigkeit und das Training wird effizienter. Die Normierungsparameter werden während des Trainings gelernt und sind für die Inferenz konstant. Weitere positive Eigenschaften der Batch-Normierung werden in [60, 105] beschrieben. Für die vorgeschlagenen

4 Dieses ließe sich zwar technisch durch Auffüllen, engl. *Padding*, an den Rändern lösen, doch hierbei würden lediglich Informationen aus dem inneren Bereich der Tensoren reproduziert werden, sodass kein zusätzlicher Informationsgewinn für die zugrunde liegende Aufgabe zu erwarten ist.

| Blockgröße | 8×8 | 16×16 | 32×32 | 64×64 |
|---|---|---|---|---|
| Eingangsfilteranzahl (EF) | 16 | 32 | 32 | 32 |
| Bottleneck-Filteranzahl (BF) | 128 | 1024 | 512 | 512 |

Tabelle 4.2: Filteranzahl für die Eingangs- und Bottleneck-Schicht

Architekturen wird die Batch-Normierung in jeder Schicht verwendet. Die beobachteten positiven Effekte sind ein schnelleres Training sowie eine reduzierte Überanpassung.

In Abhängigkeit der betrachteten Blockgröße gibt es geringe Abweichungen bei der Anzahl an Filtern in der Eingangsschicht (EF) und in der Bottleneck-Schicht (BF). Die jeweiligen Werte sind in Tabelle 4.2 gelistet.

## 4.4 TRAINING

In diesem Abschnitt wird das Training der zuvor beschriebenen Netzwerke diskutiert. Insbesondere werden hierbei das Optimierungsverfahren (der sogenannte Solver), die verwendeten Regularisierungsstrategien, das Thema Datenaugmentierung und die mit dem Training einhergehenden Hyperparameter erläutert.

Für das Training der neuronalen Netzwerke, also das Anpassen der Netzwerkkonfiguration $\mathbf{c}$ in Gleichung 2.39, wird ein stochastisches Gradientenabstiegsverfahren mit Mini-Batches, wie es in Kapitel 2.2 beschrieben wurde, verwendet. Bei dem konventionellen stochastischen Gradientenabstiegsverfahren tritt die Schwierigkeit auf, dass unabhängig von den gerade betrachteten Daten eines Mini-Batches die gleiche Lernrate für alle Parameter in der Netzwerkkonfiguration verwendet wird. Da die für den Gradientenabstieg berechneten Gradienten stark von den Daten abhängen können und auch für die verschiedenen Parameter der Netzwerkkonfiguration unterschiedlich sein können, ist es sinnvoller, die Lernrate adaptiv anzupassen.

Im Rahmen dieser Arbeit wird hierfür das Adaptive Moment Estimation (Adam)-Verfahren [56] verwendet. Bei diesem Verfahren wird anstatt des Momentums aus Gleichung 2.15 ein entsprechender Term verwendet, der auf dem ersten Moment (Mittelwert $m_i$) und dem zweiten Moment (Leistung $v_i$) der Gradienten $g_i$ zum Zeitpunkt $i$ beruht. Die Momente werden in einem gleitenden Fenster berechnet, in dem eine exponentiell

abfallende Gewichtung der vergangenen Gradienten gewählt wird. Der Aktualisierungsschritt aus Gleichung 2.38 wird erweitert zu:

$$c_{i+1} = c_i - \frac{\eta m_i}{\left(\sqrt{\frac{v_i}{1-\beta_2^i}} + \epsilon\right)(1-\beta_1^i)} \tag{4.2}$$

mit

$$m_i = \beta_1 m_{i-1} + (1-\beta_1) g_i, \tag{4.3}$$

$$v_i = \beta_2 v_{i-1} + (1-\beta_2) g_i^2. \tag{4.4}$$

Die Terme $1-\beta_1^i$ und $1-\beta_2^i$ werden verwendet, um einen Bias gegen Null zu verhindern, der entsteht, weil das gleitende Fenster mit Nullen initialisiert wird. $\beta_1$ und $\beta_2$ sind Hyperparameter, mit denen der exponentielle Abfall im gleitenden Fenster gesteuert wird. $\epsilon$ ist ein aus numerischen Gründen existierender Hyperparameter, der eine Division durch Null verhindert. Die Wahl der Werte für diese drei Hyperparameter wird später zusammen mit den anderen Hyperparametern diskutiert.

Mit der Wahl der Kostenfunktion wird festgelegt, für welches Kriterium die Konfiguration des Netzwerkes während des Trainings optimiert wird. Nur wenn die Kostenfunktion für das zu lösende Problem geeignet ist, dann kann das Training zu sinnvollen Ergebnissen führen. Typische Kostenfunktionen für die Prädiktion von (Teilen von) Bildern wären der mittlere quadratische Fehler, engl. Mean Squared Error (MSE), oder die strukturelle Ähnlichkeit, engl. Structural Similarity (SSIM), welche beide eine objektive Distanz zwischen dem prädizierten Signal und dem Originalsignal messen, sowie Funktionen, die ein subjektiv plausibles Ergebnis erzeugen. Da die vorgeschlagenen Verfahren in ein Codierungssystem eingebaut werden, das aufgrund von objektiven Kriterium die Auswahl von Codierungsverfahren durchführt, ist ein objektives Kriterium für das prädizierte Signal sinnvoll. MSE und SSIM fallen in diese Kategorie. Sie erzeugen ein prädiziertes Signal mit hohem Peak Signal-to-Noise Ratio (PSNR) (im Fall des MSE). Dieses prädizierte Signal wird jedoch niemals angezeigt. Angezeigt wird stattdessen das rekonstruierte Signal, welches sich durch Summierung des prädizierten Signals und des übertragenen Prädiktionsfehlers ergibt. Deshalb erlaubt ein hoher PSNR des prädizierten Signals keine allgemeingültige Aussage über die Videoqualität. Dieses würde lediglich eine Optimierung des PSNR des rekonstruierten Signals ermöglichen. Jedoch steht dieses Signal während des Trainings nicht zur Verfügung, da es bedingt durch die Transformationscodierung des Prädiktionsfehlers nur durch einen während des Trainings mitlaufenden Videoencoder beziehungsweise des für die Transformationscodierung mitlaufenden Submoduls ermittelt werden könnte.

Dieses ist jedoch wegen der enormen Rechenkomplexität nicht praktikabel realisierbar. Ferner hängt die Qualität des rekonstruierten Signals letzten Endes wie in Kapitel 2 beschrieben von den Rahmenbedingungen wie der zur Verfügung stehenden Datenrate und weiteren Faktoren ab. Gleichwohl ist aus Codierungssicht erstrebenswert, dass während des Trainings berücksichtigt wird, dass die Prädiktion zu einem mit geringer Datenrate übertragbaren Prädiktionsfehler führt. Ein geringer MSE des prädizierten Signals führt jedoch nicht notwendigerweise auch zu einer geringen Datenrate des zugehörigen Fehlers. Stattdessen soll die zur Codierung des Prädiktionsfehler benötigte Datenrate zumindest näherungsweise optimiert werden.

Der Fehler wird im Frequenzbereich codiert. Die durch die Transformation berechneten Koeffizienten $\mathcal{C}$ sind näherungsweise mittelwertfrei und Laplace-verteilt[5]:

$$p_{\text{coeff}}(\mathcal{C}) = \Phi e^{\Psi|\mathcal{C}|} \tag{4.5}$$

mit Konstanten $\Phi$ und $\Psi$. Die für die Codierung eines Koeffizienten benötigte Datenrate $r$ lässt sich über den Informationsgehalt $I$ annähern und ist proportional zum Betrag des Koeffizienten:

$$r \sim I\left(p_{\text{coeff}}\left(\mathcal{C}\right)\right) = -log_2\left(\Phi e^{\Psi|\mathcal{C}|}\right) \sim \Xi|\mathcal{C}| \tag{4.6}$$

mit einer Konstanten $\Xi$. Die Entropie $H$ des Prädiktionsfehlers, also eine Annäherung der für die Übertragung desselben benötigte und damit zu minimierende Datenrate unter der Bedingung einer hinreichend guten Entropiecodierung, ist damit proportional zur Summe der Absolutwerte des transfomierten Prädiktionsfehlers, engl. Sum of Absolute Transformed Differences (SATD):

$$H \sim \text{SATD} \tag{4.7}$$

Deshalb wird die SATD als Kostenfunktion gewählt.

In Tabelle 4.3 ist ein Beispiel gewählt, in dem für zwei Prädiktionsfehler die Unterschiede zwischen MSE und SATD deutlich hervortreten. Es ist ersichtlich, dass obwohl beide Prädiktionsfehler den gleichen MSE haben, die SATD deutlich unterschiedlich sind. Folglich wäre der obere Prädiktionsfehler mit einer geringeren Datenrate codierbar.

Durch den Einsatz von Regularisierungsverfahren wird eine Überanpassung auf der Trainingsdaten verhindert. Das ursprünglich von Srivastava et al. vorgeschlagene Dropout-Verfahren [113] ist ein typischer Regularisierungsansatz und wird auch in dieser Arbeit eingesetzt, um die verbleibende Überanpassung zu neutralisieren. Das Problem der Überanpassung an die Trainingsdaten von Netzwerken mit hoher Kapazität

5 Den entsprechenden Nachweis erbringt Narroschke in [88].

Tabelle 4.3: Gegenüberstellung von MSE und SATD. Beide Prädiktionfehler haben den gleichen MSE. Durch die unterschiedlichen Frequenzanteile kommt es jedoch zu stark abweichenden Koeffizienten nach der Transformation, wodurch der obere Prädiktionsfehler mit einer geringeren Datenrate codierbar ist (vgl. Gleichung 4.7).

| Prädiktionsfehler | | Koeffizienten |
|---|---|---|
| $\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$ | ⊸● | $\begin{bmatrix} 4,0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$ |
| $\mathrm{MSE} = 1,0$ | | $\mathrm{SATD} = 4,0$ |
| $\begin{bmatrix} -1 & 1 & -1 & -1 \\ -1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \end{bmatrix}$ | ⊸● | $\begin{bmatrix} -1,0 & 0 & -1,0 & 0 \\ -0,92 & 0,91 & -0,92 & -0,79 \\ 0 & -1,31 & 0 & -0,54 \\ 0,38 & 2,20 & 0,38 & -1,91 \end{bmatrix}$ |
| $\mathrm{MSE} = 1,0$ | | $\mathrm{SATD} = 12,28$ |

wird hierbei gelöst, indem in jeder Iteration nur ein zufällig bestimmter Teil des Netzwerkes trainiert wird. Die Tatsache, dass in jeder Iteration ein anderer Teil des Netzwerkes trainiert wird, lässt sich so interpretieren, dass viele Netzwerke mit geringerer Kapazität im Vergleich zum eigentlichen Netzwerk trainiert werden. Einzeln betrachtet sind diese kleinen Netzwerke nicht in der Lage, das gestellte Problem mit guter Leistung zu lösen — es kommt zu einer Unteranpassung. Für die Inferenz wird das gesamte Netzwerk eingesetzt. Dieses lässt sich so interpretieren, dass die Ergebnisse der vielen kleineren Netzwerke zu einem Gesamtergebnis kombiniert werden. Durch diese Kombination kommt es weder zu der Unteranpassung durch die Verwendung eines kleinen Netzwerkes noch zu der Überanpassung durch die Verwendung des großen, ohne Dropout trainierten, Netzwerkes.

Das klassische Dropout gemäß [113], bei dem einzelne Verbindungen entfernt werden, ist in erster Linie für vollverbundene Schichten geeignet. Würde man es auf Faltungsschichten anwenden, dann würde es durch die Korrelation zwischen örtlich benachbarten Abtastwerten laut der Erkenntnisse in [127] lediglich zu einem verlangsamten Training, nicht aber zu einer Verringerung der Überanpassung, kommen. Deshalb wird stattdessen eine Dropout-Variante eingesetzt, in der stets komplette Kanäle aus Filtern deaktiviert werden. Zur vollständigen Vermeidung von Überanpassung reicht es, wenn Dropout im Encoder eingesetzt wird. Um das Training nicht unnötig zu verlangsamen wird deshalb auf Dropout im Decoder-Teil des Netzwerkes verzichtet.

Wenn Netzwerke zu lange trainiert werden, kann es ebenfalls zu einer Überanpassung kommen. Dieses erkennt man an einer steigenden Kostenfunktion für den Validierungsdatensatz bei gleichzeitig gleichbleibenden oder weiter fallenden Werten für die Kostenfunktion für den Trainingsdatensatz. Deshalb wird immer der beste Zustand des Netzwerks auf dem Validierungsdatensatz zwischengespeichert und anschließend für die Inferenz verwendet.

Aus der Literatur ist bekannt, dass typischerweise Augmentierungsverfahren auf die Trainingsdaten angewendet werden, um vielfältigere Trainingsdaten zu erhalten. Typische Verfahren sind Spiegeln, Rotation, Helligskeits- und Kontrastanpassungen sowie das Hinzufügen von Rauschen. Da das vorliegende Problem variant gegebenüber diesen Augmentierungen ist, sind diese nicht anwendbar.

Der Trainingsvorgang hängt noch von einigen weiteren Hyperparametern ab, welche nachfolgend erläutert werden.

Durch systematische Experimente wurde eine Dropout-Rate von 30% als effizient bestimmt. Das bedeutet, dass die Wahrscheinlichkeit, dass ein Kanal eines Filters in einer Trainingsiteration nicht betrachtet wird,

als Bernoulli-Verteilung mit einer Wahrscheinlichkeit für das Nichtbetrachten von 30% modelliert wird.

Die Basis-Lernrate von Adam, also $\eta$ in Gleichung 4.2, wird auf 0.005 gesetzt. Dieser Parameter wurde durch einen Lernratentest, bei dem die Werte von $10^{-7}$ bis 1 inkrementiert wurden, bestimmt. Der Parameter $\beta_1$, der den exponentiellen Abfall im gleitenden Fenster für das erste Moment gemäß Gleichung 4.3 steuert, wird auf $0,9$ gesetzt, $\beta_2$ für das zweite Moment gemäß Gleichung 4.4 auf $0,999$ und $\epsilon$ zur Verhinderung einer Division durch Null auf $10^{-8}$. Hierbei handelt es sich um sehr typische Werte.

Die Lernrate hat wie in Kapitel 2 erläutert einen wesentlichen Einfluss auf den Erfolg des Trainings. Ist sie zu groß gewählt, dann konvergiert das Training nicht. Ist sie zu klein gewählt, dann dauert das Training zu lange beziehungsweise es findet gar kein Lernen statt. Typischerweise wird die Basis-Lernrate während des Trainings verringert, um den Gradienabstieg zu Beginn mit großen Schrittweiten durchzuführen und dann im weiteren Verlauf mit kleineren Schrittweiten in der Nähe des Minimums dieses anzunähern. Als Erweiterung hierzu wird im Rahmen dieser Arbeit eine sogenannte *One Cycle Learning Rate Policy* verwendet. Hierbei steigt die Basislernrate zunächst an, bleibt dann in der Mitte des Trainings konstant um für den letzten Teil des Trainings wieder abzufallen. Der zusätzliche Anstieg zu Beginn ist dadurch motiviert, dass die Startkonfiguration des neuronalen Netzwerkes zu Beginn des Trainings sehr weit von einem guten Arbeitspunkt entfernt sein kann. Würden die hierbei entstehenden ebenfalls sehr großen Gradienten in einem Aktualisierungsschritt mit großer Lernrate eingesetzt werden, so könnte dieses die Konfiguration des Netzwerkes in einen sehr schlechten Arbeitspunkt bewegen, aus dem heraus keine sinnvolle Optimierung mehr möglich ist. Deshalb wird zu Beginn des Trainings eine kleinere Lernrate gewählt, um zu große Sprünge der Konfiguration zu verhindern. Da der Arbeitspunkt während des ersten Teil des Trainings schnell in die Nähe einer guten Konfiguration bewegt werden kann, kann sukzessive die Lernrate erhöht werden, um im weiteren Verlauf des Trainings eine schnellere Konvergenz zu erzielen.

Die Datensätze werden zufällig in einen Trainings- und Validierungsdatensatz aufgeteilt. 90%/10% ist hierfür die gewählte Aufteilung. Die Mini-Batch-Größe wird zu 64 gewählt. Die maximale Anzahl an Epochen während des Trainings wird auf 100 gesetzt. Die Verringerung der Kostenfunktion ist zu diesem Zeitpunkt bereits sehr gering, es ist keine weitere signifikante Verbesserung zu erwarten.

# 5

# EXPERIMENTELLE UNTERSUCHUNG UND BEWERTUNG

In diesem Kapitel werden die entwickelten Verfahren zur Modellierung von Konturen und zur Abtastwertprädiktion experimentell untersucht und die hierbei erzielten Ergebnisse diskutiert. Hierfür werden die vorgeschlagenen Verfahren in den selbstentwickelten Bildcodec CoMIC sowie in das HEVC Test Model (HM), die Referenzimplementierung des HEVC-Standards, integriert. Diese beiden Implementierungen verfolgen unterschiedliche Ziele.

Die CoMIC-Implementierung ermöglicht die detaillierte Analyse der entwickelten Verfahren. Hierbei kann die Effizienz sowie der Beitrag der einzelnen Teilschritte zu dem Gesamtverfahren analysiert werden. Außerdem können Teile der entwickelten Verfahren gezielt betrachtet und zum Beispiel durch die Anpassung von Parametern optimiert werden. Die Erzielung solch eines tiefgehenden Verständnisses der Funktionsweise der entwickelten Verfahren ist bei der HM-Implementierung nicht möglich, da in einem hochoptimierten Videocodec viele aufeinanderfolgende und sich gegenseitig beeinflussende Algorithmen wie die Partitionierung in unterschiedlichen Blockrastern, die unterschiedlichen Prädiktionsverfahren, die Rate-Distortion-Optimierung, hocheffiziente Entropiecodierung durch CABAC, Nachverarbeitungsverfahren wie Sample Adaptive Offset und Deblockingfilter und so weiter verwendet werden. Durch das Hinzufügen eines neuen Verfahrens gäbe es Auswirkungen auf die Verwendung aller anderen bereits existierenden Verfahren.

Die Implementierung in einem Videocodec wie HM hingegen ermöglicht die Bewertung der Codiereffizienz der entwickelten Verfahren in einer realen Videocodierungsanwendung und somit den Vergleich mit dem aktuellen Stand der Technik. Zusätzlich lässt sich mit dieser Implementierung eine Optimierung der Übertragung von Seiteninformationen im Hinblick auf die verwendete Entropiecodierung mit CABAC vornehmen.

## 5.1 INTEGRATION IN EINEN BILDCODEC

Das Ziel dieses Teilexperiments ist die Evaluierung der vorgeschlagenen Verfahren unter Laborbedingungen. Hierfür wird ein selbstentwickelter Bildcodec namens CoMIC, welcher in der Programmiersprache C++ implementiert wurde, verwendet.

Das Bild wird zu Beginn der Codierung in gleichgroße Blöcke aufgeteilt. Hierdurch entfällt die Abhängigkeit der Prädiktion von den vielfältigen Partitionierungsmöglichkeiten, welche es in HEVC gibt[1]. Im Rahmen der Evaluierung des CoMIC-Codecs werden die Blockgrößen $8 \times 8$, $16 \times 16$, $32 \times 32$ und $64 \times 64$ verwendet. Jeder Block wird mit den gleichen Verfahren codiert. Hierfür wird vor Beginn der Codierung festgelegt, welches Verfahren zur Konturmodellierung (polynomiell oder stochastisch) und welches Verfahren zur Abtastwertprädiktion (*Along-contour* oder neuronale Netzwerke) verwendet werden sollen. Dieses entspricht dem Setzen der beiden Schalter in Abbildung 3.5. Durch die Anwendung der ausgewählten Verfahren wird eine Prädiktion für den zu codierenden Block erzeugt. Eine Modeauswahl findet anders als im Videocodec nicht statt, da die zu verwendenden Codierungsverfahren vor Beginn der Codierung festgelegt wurden. Zur Übertragung des Prädiktionsfehlers wird dieser mit einer DCT in den Frequenzbereich transformiert. Die hierbei entstehenden Prädiktionsfehlerkoeffizienten werden quantisiert. Hierfür wird eine gleichförmige Quantisierung durch eine Division und Multiplikation mit einem QP realisiert. Eine Signalisierung von zusätzlichen Syntaxelementen zur Übertragung der Modeentscheidungen ist ebenfalls anders als im Videocodec nicht erforderlich.

Nachfolgend werden die für die Experimente mit dem CoMIC-Codec verwendeten Metriken erläutert. Die Datenrate wird über die Entropie der quantisierten Prädiktionsfehlerkoeffizienten angenähert. Die Entscheidung hierfür und gegen die Verwendung der durch eine Entropiecodierung der quantisierten Prädiktionsfehlerkoeffizienten erreichten realen Datenrate ist wie folgt motiviert: Die umfangreichen Bemühungen von Entropiecodierungsverfahren, vorhergehenden Binarisierungsverfahren sowie komplexen Syntaxkonstrukten zur Beschreibung der Prädiktionsfehlerkoeffizienten dient dazu, diese mit einer Datenrate möglichst nah an der Entropie der zu codierenden Werte zu codieren. Nun ließe sich entweder ein geeignetes Verfahren entwickeln oder ein vorhandenes Verfahren wie CABAC auswählen. Jedoch wäre es nicht notwendigerweise zu erwarten, dass es sich hierbei um das bestmögliche Entropiecodie-

1 Es ist zu beachten, dass die hiermit erzielte Codierungseffizienz lediglich einer Untersuchung unter Laborbedingungen dient. Die Codierungseffizienz ist deutlich geringer als in modernen Videocodecs mit solchen vielfältigen Partitionierungsmöglichkeiten. Die Codierungseffizienz unter den Bedingungen solch eines Videocodecs wird in Abschnitt 5.2 untersucht.

rungsverfahren handelt, was auch gar nicht Kern des durchgeführten Experiments wäre.

Alternativ könnte die Prämisse vertreten werden, dass ein entsprechendes Verfahren existiere, ohne es zu implementieren. Die Entropie ist die untere Grenze für die mit einer hinreichend guten Entropiecodierung bei einer verlustlosen[2] Codierung erreichbaren Datenrate. Da im nachfolgenden Abschnitt 5.2 für die Evaluierung der vorgeschlagenen Verfahren in einem Videocodec ohnehin die Codierung des entstehenden Prädiktionsfehlers mit CABAC verwendet wird, erscheint die Beschränkung in diesem Abschnitt auf die Datenratenannäherung per Entropie sinnvoll und vertretbar.

Die Entropie wird berechnet, indem zunächst für die Koeffizienten des codierten Blocks ein Histogramm für die Werte der Transformationskoeffizienten erstellt wird. Basierend auf diesem Histogramm werden die relativen Häufigkeiten $p_i$ der unterschiedlichen Koeffizienten berechnet. Mit diesen relativen Häufigkeiten ergibt sich die Entropie $H$ für einen Block der Größe $s \times s$ zu[3]:

$$H = \sum_{i=1}^{s \times s} -p_i log_2(p_i). \tag{5.1}$$

Der MSE wird gemessen und hieraus wird das PSNR berechnet:

$$\mathrm{PSNR} = 10 \cdot \log_{10}\left(\frac{S_{\max}^2}{\mathrm{MSE}}\right), \tag{5.2}$$

mit dem maximalen Signalwert $S_{\max}$, also $S_{\max} = 255$ für 8 Bit Videosignale.

Die Encoderlaufzeit wird als Prozessor-, engl. Central Processing Unit (CPU)-, Zeit[4] gemessen. Die Messung der Decoderlaufzeit entfällt, da es keine Modeauswahl gibt. Hierdurch unterscheidet sich die Laufzeit zwischen Encoder und Decoder nicht.

2 Die als verlustlos angenommene Entropiecodierung impliziert nicht, dass die komplette Codierung verlustlos ist. Durch den entstehenden Quantisierungsfehler ist die Bildcodierung verlustbehaftet.

3 Es ist zu beachten, dass hier eine Annahme über die Adaptierungsfähigkeit des Entropiecodierungsverfahrens zu Grunde gelegt wird, die für realen Entropiecodierungsverfahren möglicherweise nur näherungsweise zutrifft.

4 Da zum Teil mehrere Prozesse gleichzeitig für die Berechnung der implementierten Verfahren verwendet werden, ist hier eine Unterscheidung zwischen der Laufzeit des Programms und der hierbei verwendeten CPU-Zeit notwendig. Wird nur ein Prozess gleichzeitig verwendet, so sind die beiden Zahlenwerte identisch. Werden mehrere Prozesse gleichzeitig verwendet, dann ist die als Summe der Laufzeiten der einzelnen Prozesse berechnete CPU-Zeit entsprechend größer. Da nur die CPU-Zeit den tatsächlichen Rechenaufwand der Verfahren erfasst, wird diese verwendet.

Zunächst wird der Experimentaufbau beschrieben. Als Eingangsdaten dienen Bilder aus zwei populären Datenbanken, die häufig für die Evaluierung von Bildcodierungsverfahren verwendet werden: die Kodak Image Dataset (KODIM) und die University of Southern California - Signal and Information Processing Institute (USC-SIPI)-Bilddatenbank. Keine dieser Datenbanken wurde für das Training der neuronalen Netzwerke oder die Optimierung anderer Hyperparameter der vorgeschlagenen Verfahren verwendet.

Bei den 24 Bildern der KODIM, welche Auflösungen von $768 \times 512$ und $512 \times 768$ haben, handelt es sich um klassische Fotografiemotive [79]. Obgleich es sich bei den Bildern aus diesen Datenbank um die typischerweise für die Evaluierung von Bildcodierungsverfahren verwendeten Bilder handelt, ist anzumerken, dass die Bildeigenschaften nicht in allen Fällen repräsentativ für moderne Digitalkameras sind.

In der USC-SIPI-Datenbank gibt es 141 Bilder[5]. Diese haben Auflösungen von $256 \times 256$, $512 \times 512$ und $1024 \times 1024$. Die Bilder sind aufgeteilt in die Kategorien *Textures* (Nahaufnahmen von Texturen), *Aerials* (Luftaufnahmen) und *Miscellaneous* (klassische Fotografiemotive). [135]

Einige exemplarische Bilder aus den Datenbanken sind in Abbildung 5.1 dargestellt.

5 Es werden alle Bilder bis auf drei (Tiffany, Lena, Elaine) verwendet. Die Herausgeber der Datenbank vertreten den Standpunkt, dass die Verwendung dieser Bilder aus ethischen Gründen nicht mehr zeitgemäß sei und bieten deshalb die Dateien seit 2018 nicht mehr zum Herunterladen an [136].


Abbildung 5.1: Exemplarische Darstellung einiger Testbilder

Es werden die folgenden Vergleiche durchgeführt:

1. Bestimmung des Mehrwerts des vorgeschlagenen Konturmodells

   Testverfahren: Stochastische Konturmodellierung mit *Along-contour*-Abtastwertprädiktion

   Referenzverfahren: Polynomielle Konturmodellierung mit *Along-contour*-Abtastwertprädiktion

2. Bestimmung des Mehrwerts des vorgeschlagenen Verfahrens zur Abtastwertprädiktion

   Testverfahren: Polynomielle Konturmodellierung mit neuronalen Netzwerken als Abtastwertprädiktor

   Referenzverfahren: Polynomielle Konturmodellierung mit *Along-contour*-Abtastwertprädiktion

In beiden Vergleichen wird das polynomielle Konturmodell in Kombination mit der *Along-contour*-Abtastwertprädiktion verwendet, da in den eigenen Vorarbeiten bereits gezeigt wurde, dass hierdurch die Codierungseffizienz von HEVC verbessert werden kann.

Es werden Quantisierungsparameter im Bereich 1 bis 31 mit einem Abstand von 5 festgelegt. Für die Durchführung der Simulationen wurde eine homogene Serverinfrastruktur mit Intel Xeon Gold 5120 CPUs verwendet. Die Inferenz der neuronalen Netzwerke wurde ebenfalls auf der CPU berechnet. Potentiell ließe sich die Inferenz durch eine GPU-Beschleunigung schneller berechnen. Dieser Aspekt wird im Rahmen dieser Arbeit nicht weiter betrachtet.

### 5.1.1 *Mehrwert des vorgeschlagenen Konturmodells*

In diesem Teilexperiment wird der Mehrwert des vorgeschlagenen Konturmodells gegenüber dem polynomiellen Konturmodell aus der eigenen Vorarbeit [69] evaluiert. Hierfür werden die gemessenen BD-Raten, das BD-PSNR sowie die Erhöhung der Rechenkomplexität betrachtet. Die BD-Raten und das BD-PSNR werden gemäß [7, 8] berechnet und nachfolgend kurz anhand von Abbildung 5.2 skizziert. Der Beginn der Berechnung ist für beide Metriken gleich. Sowohl für das Testverfahren als auch für das Referenzverfahren werden die gemessenen Datenpunkte bestehend aus Datenrate und PSNR als RD-Kurve aufgetragen. Zwischen den Datenpunkten der jeweiligen Verfahren wird mittels stückweise kubischer Splines interpoliert. Über die beiden Kurven wird nun integriert und die Differenz, deshalb Bjøntegaard-*Delta*, der Flächen berechnet. Ab diesem Schritt unterscheiden sich die beiden Metriken.

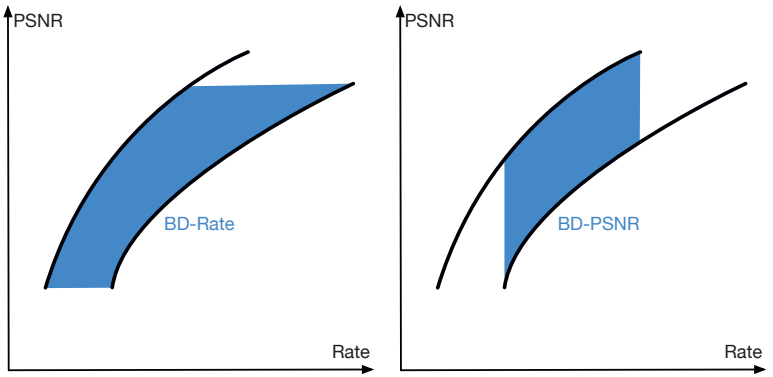



Abbildung 5.2: Veranschaulichung der Berechnung von BD-Rate (links) und BD-PSNR (rechts). Die BD-Rate misst die Datenratenveränderung bei gleichbleibender Bildqualität. Dementsprechend impliziert ein negativer Wert eine verbesserte Codierungseffizienz. Das BD-PSNR gibt die Qualitätsveränderung bei gleichbleibender Datenrate an. Ein positiver Wert zeigt eine gesteigerte Codierungseffizienz an.

Mit der BD-Rate soll die Veränderung der Datenrate bei gleicher Qualität gemessen werden. Hierfür wird der Bereich zwischen den beiden RD-Kurven durch zwei zur Raten-Achse parallele Linien begrenzt und jeweils zwischen diesen Grenzen integriert. Ein negativer Zahlenwert bedeutet eine Verringerung der Datenrate bei gleichbleibender Qualität und somit eine Verbesserung der Codierungseffizienz durch das Testverfahren im Vergleich zum Referenzverfahren. Mit dem BD-PSNR wird hingegen die Veränderung des PSNR bei gleichbleibender Datenrate gemessen. Entsprechend werden die Grenzen und die Integrale wie in dem rechten Teil der Abbildung 5.2 veranschaulicht bestimmt. Ein positiver Zahlenwert bedeutet einen gestiegenen PSNR-Wert bei gleichbleibender Datenrate, also eine Verbesserung der Codierungseffizienz. Da das PSNR in Dezibel angegeben wird, wird das gleiche für das BD-PSNR umgesetzt. Durch den abweichenden Integrationsbereich, der vom Verlauf der RD-Kurven abhängt, lässt sich keine triviale Rechenvorschrift zur direkten Umwandlung von BD-Raten in BD-PSNR-Werte angeben.

Die Komplexitätssteigerung wird als Quotient aus der Laufzeit des Testverfahrens und des Referenzverfahrens ermittelt.

Die berechneten Metriken sind in Tabelle 5.1 zusammengefasst. Ferner sind die BD-Raten und die Komplexitätssteigerung in den Abbildungen 5.3 und 5.4 veranschaulicht. Es ist zu beobachten, dass die mittleren Codierungsgewinne durch den Einsatz des vorgeschlagenen Konturmo-

Tabelle 5.1: Codierungsergebnisse für die CoMIC-Implementierung. Testverfahren: Stochastische Konturmodellierung mit *Along-contour*-Abtastwertprädiktion. Referenzverfahren: Polynomielle Konturmodellierung mit *Along-contour*-Abtastwertprädiktion. Negative BD-Raten sowie positive BD-PSNR-Werte zeigen eine gesteigerte Codierungseffizienz an. Die Komplexität ist als Quotient der Laufzeiten des Testverfahrens und des Referenzverfahrens angegeben.

| Blockgröße | Bilddatenbank | BD-Rate | BD-PSNR [dB] | Komplexität |
|---|---|---|---|---|
| 8x8 | Textures | -0.5% | 0.08 | 73 |
| | Aerials | 0.0% | 0.01 | 65 |
| | Miscellaneous | -0.3% | 0.05 | 110 |
| | Kodim | -0.1% | 0.02 | 24 |
| | **Gesamt** | **-0.2%** | **0.00** | **80** |
| 16x16 | Textures | -0.5% | 0.08 | 60 |
| | Aerials | -0.2% | 0.03 | 46 |
| | Miscellaneous | -1.0% | 0.14 | 74 |
| | Kodim | -0.9% | 0.14 | 28 |
| | **Gesamt** | **-0.7%** | **0.11** | **59** |
| 32x32 | Textures | -1.6% | 0.23 | 150 |
| | Aerials | -0.6% | 0.06 | 45 |
| | Miscellaneous | -2.1% | 0.29 | 74 |
| | Kodim | -1.7% | 0.27 | 62 |
| | **Gesamt** | **-1.6%** | **0.23** | **88** |
| 64x64 | Textures | -1.9% | 0.26 | 1078 |
| | Aerials | -1.1% | 0.12 | 58 |
| | Miscellaneous | -2.1% | 0.32 | 97 |
| | Kodim | -2.1% | 0.34 | 111 |
| | **Gesamt** | **-1.9%** | **0.26** | **408** |

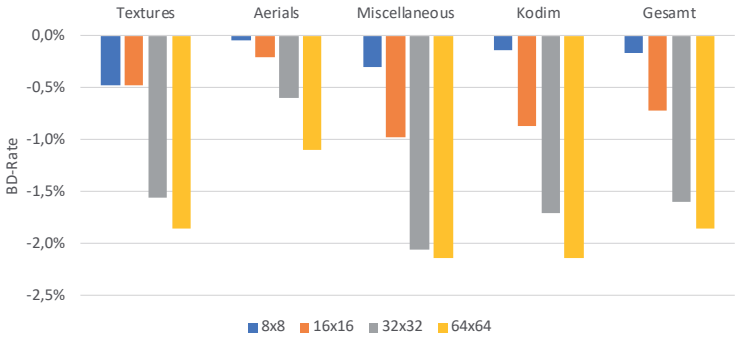



Abbildung 5.3: BD-Rate für die *CoMIC*-Implementierung. Testverfahren: Stochastische Konturmodellierung mit *Along-contour*-Abtastwertprädiktion. Referenzverfahren: Polynomielle Konturmodellierung mit *Along-contour*-Abtastwertprädiktion. Negative Zahlenwerte zeigen eine gesteigerte Codierungseffizienz an.

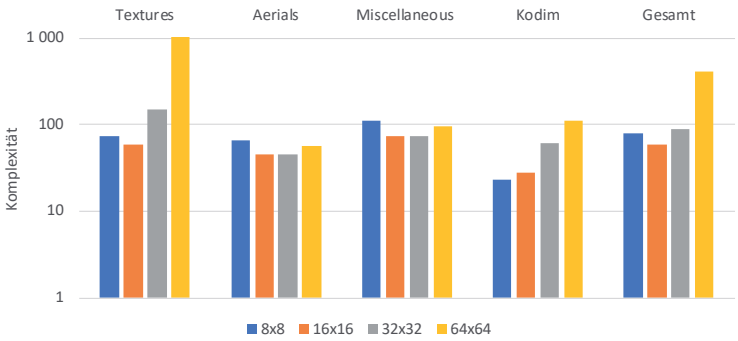



Abbildung 5.4: Komplexitätssteigerung für die *CoMIC*-Implementierung. Testverfahren: Stochastische Konturmodellierung mit *Along-contour*-Abtastwertprädiktion. Referenzverfahren: Polynomielle Konturmodellierung mit *Along-contour*-Abtastwertprädiktion. Die Komplexität ist als Quotient der Laufzeiten des Testverfahrens und des Referenzverfahrens angegeben.

dells gemittelt über alle Testbilder mit der Blockgröße steigen. Für Blöcke der Größe $64 \times 64$ werden die größten Gewinne mit einer BD-Rate von $-1,9\%$ gemessen. Die Korrelation zwischen Blockgröße und Codierungseffizienz lässt sich in diesem Fall dadurch erklären, dass der Basiswert für die Berechnung der Codierungseffizienz ebenfalls auf einer Konturmodellierung, der weniger genauen polynomiellen Konturmodellierung, beruht. Der bei der Extrapolation entstehende größere Fehler für das polynomielle Konturmodell wird umso größer, je weiter die Kontur extrapoliert wird. Des Weiteren kann beobachtet werden, dass die Effizienzsteigerung von den zu codierenden Inhalten abhängt. Während die Verfahren für klassische Fotografiemotive (*KODIM* und *Miscellaneous*) und Texturen gut funktionieren, liegen die Werte für Luftbildaufnahmen (*Aerials*) jeweils etwas niedriger. Die Steigerung der Komplexität liegt in Abhängigkeit der Blockgröße zwischen 59 und 408. Hierbei ist bemerkenswert, dass die Komplexität tendenziell mit der Blockgröße und bei Inhalten mit vielen Konturen (*Textures*) steigt.

### 5.1.2 *Mehrwert der vorgeschlagenen Abtastwertprädiktion*

Die Auswertung dieses Teilexperiments zur Bewertung des Mehrwerts des vorgeschlagenen Verfahrens zur Abtastwertprädiktion mittels neuronaler Netzwerke gegenüber dem *Along-contour*-Referenzverfahren aus den eigenen Vorarbeiten [65] und [69] erfolgt analog zum im vorherigen Abschnitt beschriebenen Teilexperiment. Die neuronalen Netzwerke wurden wie in Kapitel 4 beschrieben trainiert. Es gibt keine Überlappung zwischen den für das Training verwendeten Datenbanken und den für die Evaluierung verwendeten Datenbanken. Die Codierungsergebnisse sind in Tabelle 5.2 zusammengefasst. Die BD-Raten und die Komplexitätssteigerung sind in den Abbildungen 5.5 und 5.6 visualisiert.

Tabelle 5.2: Codierungsergebnisse für die CoMIC-Implementierung. Testverfahren: Polynomielles Konturmodellierung mit neuronalen Netzwerken als Abtastwertprädiktor. Referenzverfahren: Polynomielle Konturmodellierung mit *Along-contour*-Abtastwertprädiktion. Negative BD-Raten sowie positive BD-PSNR-Werte zeigen eine gesteigerte Codierungseffizienz an. Die Komplexität ist als Quotient der Laufzeiten des Testverfahrens und des Referenzverfahrens angegeben.

| Blockgröße | Bilddatenbank | BD-Rate | BD-PSNR [dB] | Komplexität |
|---|---|---|---|---|
| 8x8 | Textures | -3.2% | 0.44 | 7.7 |
| | Aerials | -2.3% | 0.31 | 8.9 |
| | Miscellaneous | -3.6% | 0.53 | 9.2 |
| | Kodim | -3.5% | 0.58 | 8.5 |
| | **Gesamt** | **-3.5%** | **0.47** | **8.7** |
| 16x16 | Textures | -6.1% | 0.86 | 22.0 |
| | Aerials | -5.9% | 0.76 | 25.4 |
| | Miscellaneous | -7.8% | 1.12 | 29.2 |
| | Kodim | -7.2% | 1.11 | 23.9 |
| | **Gesamt** | **-7.2%** | **1.00** | **26.2** |
| 32x32 | Textures | -7.4% | 1.11 | 23.8 |
| | Aerials | -6.7% | 0.85 | 27.4 |
| | Miscellaneous | -9.9% | 1.42 | 30.7 |
| | Kodim | -8.8% | 1.35 | 26.4 |
| | **Gesamt** | **-8.8%** | **1.23** | **28.3** |
| 64x64 | Textures | -6.7% | 1.04 | 25.2 |
| | Aerials | -5.3% | 0.70 | 30.6 |
| | Miscellaneous | -7.9% | 1.24 | 29.9 |
| | Kodim | -7.3% | 1.10 | 28.7 |
| | **Gesamt** | **-6.6%** | **1.02** | **29.2** |

In Abhängigkeit der Blockgröße liegen die mittleren BD-Raten zwischen $-3,5\%$ für $8 \times 8$-Blöcke und $-8,8\%$ für $32 \times 32$-Blöcke. Für $64 \times 64$-Blöcke sind die Codierungsgewinne etwas geringer als für die Blockgrößen $16 \times 16$ und $32 \times 32$. Für jede Blockgröße lässt sich beobachten, dass die größten Codierungsgewinne für die Kategorien *Miscellaneous*

und *KODIM*, in denen klassischen Fotografiemotive zu finden sind, am größten sind. Dieses lässt sich damit erklären, dass diese Inhalte auch in der für das Training der neuronalen Netzwerke verwendeten Datenbank abgebildet sind. Die Bilder mit den spezielleren Inhalten in den Kategorien *Textures* und *Aerials*, die nicht Bestandteil der Trainingsdatenbank sind, führen zu geringeren Gewinnen. Dass trotzdem Gewinne messbar sind erschließt sich dennoch leicht, da beispielsweise auch in normalen Fotografien Texturen vorkommen. Die Komplexitätssteigerung liegt für die meisten Blockgrößen zwischen 20 und 30, für die kleinste Blockgröße unter 10.

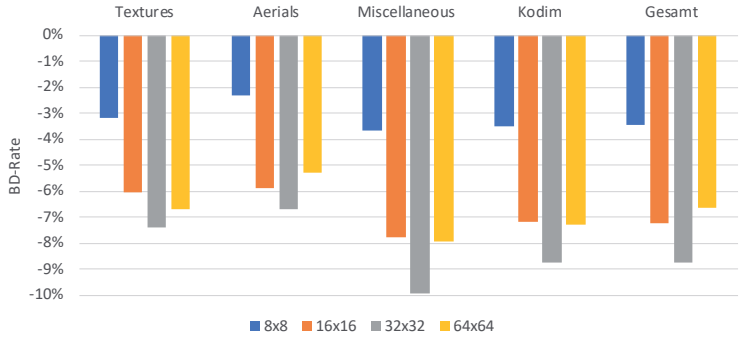



Abbildung 5.5: BD-Rate für die *CoMIC*-Implementierung. Testverfahren: Polynomielles Konturmodellierung mit neuronalen Netzwerken als Abtastwertprädiktor. Referenzverfahren: Polynomielle Konturmodellierung mit *Along-contour*-Abtastwertprädiktion. Negative Zahlenwerte zeigen eine gesteigerte Codierungseffizienz an.

Es konnte gezeigt werden, dass die vorgeschlagenen Verfahren die korrespondierenden Verfahren aus den Vorarbeiten jeweils verbessern. Somit ist die Kombination der beiden vorgeschlagenen Verfahren zu einem Intra-Codierungsverfahren erfolgsversprechend. Noch nicht berücksichtigt wurde in den bisherigen Experimenten, dass dieses Intra-Codierungsverfahren in realen Videocodecs mit anderen Codierungsverfahren kombiniert würde. Die Erkenntnis, wie sich die Codierungseffizienz in Kombination mit den existierenden Verfahren verhält, kann nur durch die Integration des vorgeschlagenen Intra-Codierungsverfahrens in einen Videocodec erlangt werden. Das entsprechende Experiment wird in Abschnitt 5.2 geschildert.

Einige exemplarische Beispiele für mit den vorgeschlagenen Verfahren erzeugte Prädiktionssignale sind in Abbildung 5.7 visualisiert. In der oberen Reihe ist jeweils das Originalsignal zu dem prädizierten Signal in

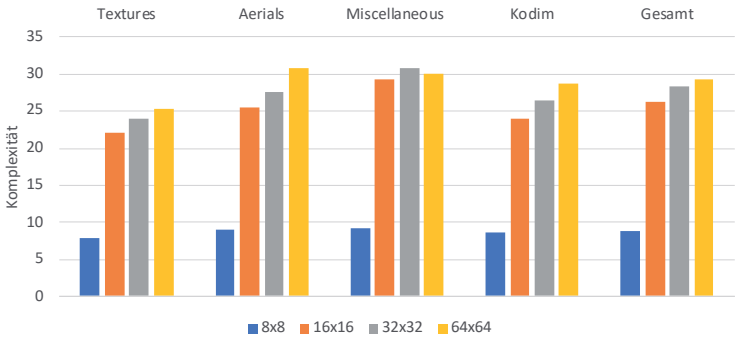



Abbildung 5.6: Komplexitätssteigerung für die *CoMIC*-Implementierung. Testverfahren: Testverfahren: Polynomielles Konturmodellierung mit neuronalen Netzwerken als Abtastwertprädiktor. Referenzverfahren: Polynomielle Konturmodellierung mit *Along-contour*-Abtastwertprädiktion. Die Komplexität ist als Quotient der Laufzeiten des Testverfahrens und des Referenzverfahrens angegeben.

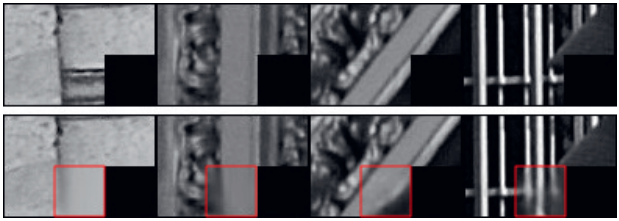


Abbildung 5.7: Beispiele für die Prädiktion mit dem vorgeschlagenen Verfahren. Obere Reihe: Originalsignal, untere Reihe: Prädiktion.

der unteren Reihe gezeigt. Die Beispiele zeigen sowohl die Limitierungen der Verfahren als auch Fälle in denen die Prädiktion gut funktioniert. Im ersten Beispiel wird das Signal im Bereich des zu codierenden Blocks dunkel, im letzten Beispiel kann nicht prädiziert werden, dass die Gitterstäbe teilweise hintereinander liegen. Beides ist aus dem Referenzbereich nicht ersichtlich. Mit dem Beispiel ganz rechts kann gezeigt werden, dass auch Blöcke mit mehreren Konturen in unterschiedlichen Richtungen zuverlässig prädiziert werden können. Weitere Beispiele für prädizierte Signale sind in Abbildung 5.8 dargestellt.

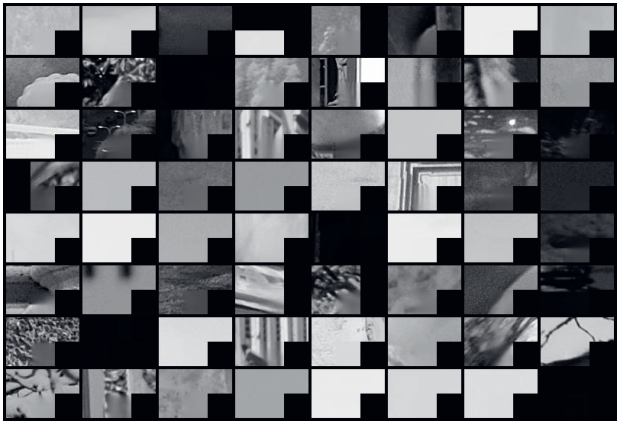


Abbildung 5.8: Weitere Prädiktionsbeispiele

### 5.1.3 *Einordnung von Codierungseffizienz und Komplexitätssteigerung*

Der Vorschlag neuer Codierungsverfahren erfordert eine Abwägung zwischen der Steigerung der Codierungseffizienz und der hiermit einhergehenden Steigerung des Rechenaufwands. In der Regel ist es durch das Einführen zusätzlicher Codierungsverfahren nicht möglich, die Codierungseffizienz zu steigern und gleichzeitig den Rechenaufwand zu senken. Somit ist zu bewerten, ob der zusätzliche Rechenaufwand für die Steigerung der Codierungseffizienz vertretbar ist. In Abbildung 5.9 sind für die unterschiedlichen Experimente, das heißt die Vergleiche der Konturmodelle sowie der Abtastwertprädiktoren, jeweils für vier Blockgrößen die BD-Raten über der Steigerung des Rechenaufwands aufgetragen. Die Komplexitätssteigerung wird hierbei, wie auch im vorherigen Abschnitt, als Quotient aus der Laufzeit des Testverfahrens und der Laufzeit des Referenzverfahrens gemessen. Die Datenpunkte der jeweiligen Datenreihen für den Vergleich der Abtastwertprädiktoren sind tendenziell vertikal verteilt während die Datenpunkte für den Vergleich der Konturmodelle eine starke horizontale Spreizung aufweisen.

Der Rechenaufwand für die Inferenz mit den neuronalen Netzwerken ist unabhängig von dem als Eingang anliegenden Signal. Deshalb ist es leicht ersichtlich, dass die zu einer Blockgröße gehörenden Datenpunkte mit einem sehr ähnlichen Rechenaufwand codiert wurden. Es ist ebenfalls zu beobachten, dass die Komplexität für die Blockgrößen $16 \times 16$ bis $64 \times 64$ in einer ähnlichen Größenordnung ist während sie für $8 \times 8$-Blöcke geringer ist. Dieses lässt sich über die unterschiedlichen

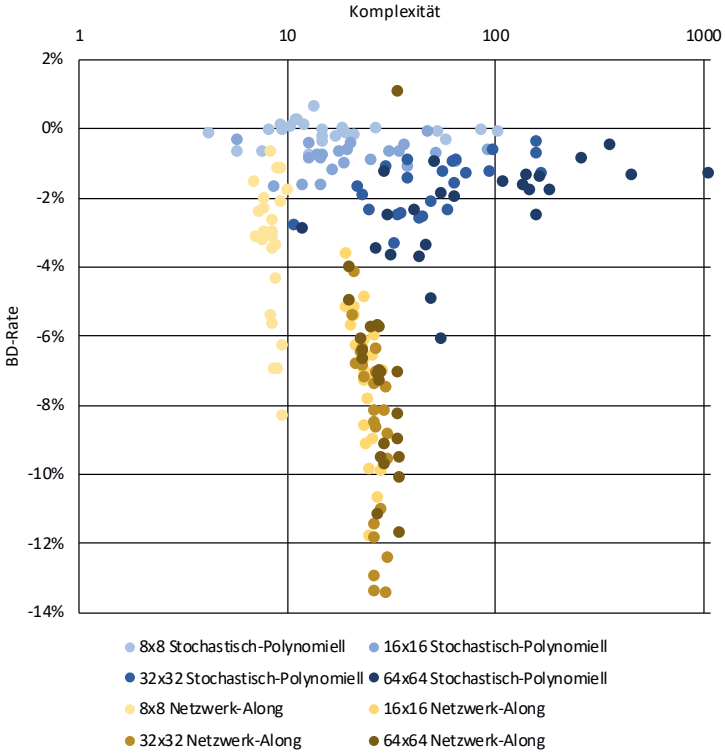



Abbildung 5.9: Einordnung von Verbesserung der Codierungseffizienz und hiermit einhergehender Komplexitätssteigerung für die CoMIC-Implementierung. Syntax für Legendeneinträge: Blockgröße Testverfahren-Referenzverfahren. Negative BD-Raten zeigen eine gesteigerte Codierungseffizienz an. Die Komplexität ist als Quotient der Laufzeiten des Testverfahrens und des Referenzverfahrens angegeben.

Filter- und Schichtanzahlen der Netzwerke erklären. Die Qualität der Prädiktion durch das neuronale Netzwerk hingegen hängt sehr wohl von dem Eingangssignal ab. Hierdurch ist es erklärbar, dass die Steigerung der Codierungseffizienz gemessen als BD-Rate für die unterschiedlichen Bilder schwankt.

Für den Vergleich der beiden Konturmodelle sind die Schwankungen innerhalb der Datenreihen für die unterschiedlichen Blockgrößen größer. Dieses lässt sich hiermit begründen, dass die Komplexität der

stochastischen Konturmodellierung durch die für die Optimierung der Hyperparameter der Kovarianzfunktion notwendige Matrixinvertierung gemäß Gleichung 3.14 kubisch mit der Anzahl an Konturpixeln $N$ ansteigt, $\mathcal{O}\left(N^3\right)$. Die Konturlängen können für verschiedene Bilder unterschiedlich sein und steigen tendenziell mit der Blockgröße. Dieses lässt sich ebenfalls in den Messwerten beobachten.

Die vorgeschlagenen Verfahren verursachen eine nicht zu vernachlässigende Steigerung der Rechenkomplexität im Vergleich zu den Referenzverfahren. Es ist daher von Interesse zu ergründen, welche Teilverfahren für die gesteigerte Komplexität verantwortlich sind. Hierfür wurde eine detaillierte Analyse des Laufzeitverhaltens der Software mit einem sogenannten *Profiler* durchgeführt. Mit einem geeigneten Profiler lässt sich messen, wie viel Rechenzeit jede Zeile des Programmcodes für die Ausführung benötigt. Es wurde der Profiler *Callgrind* verwendet. Die Berechnungen erfolgten auf einer Intel Core i9-9900K CPU. Es wird für die beiden in den Abschnitten 5.1.1 und 5.1.2 geschilderten Experimente das gleiche Bild mit den gleichen Parametern codiert. Die wesentlichen Ergebnisse der Analyse werden im Folgenden näher ausgeführt.

In Abbildung 5.10 sind die Ergebnisse für die Codierung mit dem polynomiellen Konturmodell und dem neuronalen Netzwerk als Abtastwertprädiktor veranschaulicht. Es lässt sich schlussfolgern, dass die Abtastwertprädiktion die Rechenkomplexität dominiert: Die Inferenz mit dem neuronalen Netzwerk ist für 93,9% der Rechenzeit verantwortlich. Die weiteren nennenswerten Teilverfahren mit Bezug auf die Rechenkomplexität sind die Konturdetektion mit 4,5%, die Konturvorverarbeitung mit 0,1% sowie die Konturmodellierung mit dem polynomiellen Konturmodell mit 0,7% der Rechenzeit. Die restlichen Teilverfahren sind für die verbleibenden 0,8% der Rechenzeit verantwortlich.

Die Ergebnisse für das Experiment mit dem stochastischen Konturmodell in Kombination mit dem neuronalen Netzwerk sind in Abbildung 5.11 visualisiert. Es ist ersichtlich, dass in diesem Fall die Rechenkomplexität durch die Konturmodellierung dominiert wird. Ein Großteil der Rechenzeit, 95,4%, lassen sich der Cholesky-Zerlegung gemäß Gleichung 3.14 zuordnen. Die Konturdetektion kommt auf 0,1%, die Inferenz mit dem neuronalen Netzwerk auf 3%, die Konturextrapolation mit dem Gauß-Prozess auf 0,2%. Die sonstigen Teilschritte verursachen zusammen 1,3% der Rechenzeit.

Zusammenfassend lässt sich festhalten, dass die Inferenz mit dem neuronalen Netzwerk sowie die Cholesky-Zerlegung für einen Großteil der Rechenzeit verantwortlich sind. Es ist anzunehmen, dass diese durch weitergehende Optimierung potentiell beschleunigt werden können. Die Inferenz könnte durch die sehr reguläre Architektur von neuronalen

Netzwerken durch eine geeignete Hardwarebeschleunigung in kommerziellen Implementierungen optimiert werden. Die Cholesky-Zerlegung könnte höchstwahrscheinlich durch eine hochoptimierte Implementierung oder durch eine Annäherung mit geeigneten Verfahren beträchtlich beschleunigt werden.

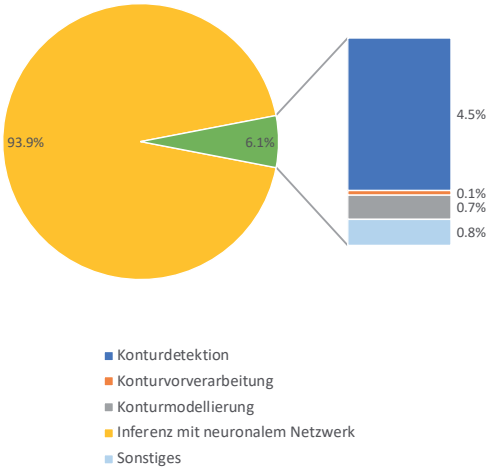



Abbildung 5.10: Analyse der Rechenkomplexität für das polynomielle Konturmodell mit neuronalem Netzwerk als Abtastwertprädiktor

Abschließend erscheint es sinnvoll festzuhalten, dass bei der Einordnung der Codierungsgewinne zu berücksichtigen ist, dass sich diese auf einen Versuch mit dem selbstentwickelten Bildcodec CoMIC beziehen, in dem als Referenzverfahren lediglich die eigenen Vorarbeiten verwendet werden. Dieses Experiment ermöglicht lediglich eine Einordnung der vorgeschlagenen Verfahren mit Bezug auf die Verfahren aus der eigenen Vorarbeit. Um eine Einordnung der Codierungseffizienz der vorgeschlagenen Verfahren mit dem Stand der Technik zu ermöglichen, ist dieses Experiment ungeeignet. Hierfür ist eine Integration der vorgeschlagenen Verfahren in einen Videocodec notwendig. Das entsprechende Experiment wird im restlichen Verlauf dieses Kapitels erörtert.

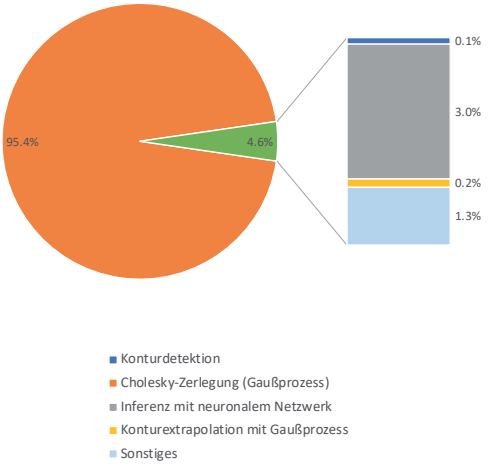



Abbildung 5.11: Analyse der Rechenkomplexität für das stochastische Konturmodell mit neuronalem Netzwerk als Abtastwertprädiktor

## 5.2 INTEGRATION IN EINEN VIDEOCODEC

Zunächst wird die Implementierung in der HM-Software, hierbei handelt es sich um die in der Programmiersprache C++ geschriebene Referenzimplementierung des HEVC-Standards, anhand von Abbildung 5.12 erläutert. Die vorgeschlagenen Verfahren werden als zusätzliches Codierungsverfahren parallel zu den existierenden Codierungsverfahren implementiert. Die zu codierenden Bilder werden gemäß des in Kapitel 2 beschriebenen Partitionierungsverfahrens in Blöcke aufgeteilt. Die vorgeschlagenen Verfahren werden für jede Prädiktionstiefe auf CU-Ebene getestet. Hierfür wird eine Prädiktion erzeugt, in dem zuerst das Verfahren zur Konturmodellierung und dann das Verfahren zur Abtastwertprädiktion eingesetzt werden. Der bei der Prädiktion entstehende Prädiktionsfehler wird mit dem ebenfalls in Kapitel 2 beschriebenen Verfahren transformationscodiert. Hierbei kann der Fehlerblock durch den TU-Baum weiter partitioniert werden.

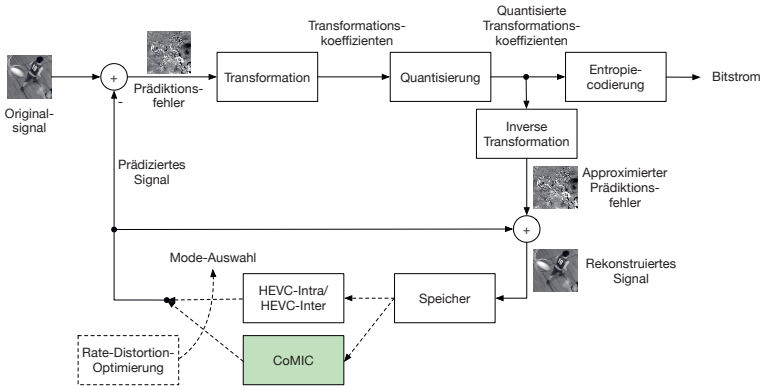



Abbildung 5.12: Integration der vorgeschlagenen Verfahren in einen Videocodec

Die Rate-Distortion-Optimierung wird im Encoder dazu genutzt, um mit der Encodersteuerung diejenige Kombination aus Partitionierung, Modeauswahl und Parameterwahl für die Modi zu finden, welche die geringsten RD-Kosten verursacht. Das CoMIC-Verfahren wird ausgewählt, wenn es in Kombination mit einer gefundenen Partitionierung für eine CU die beste Wahl mit Bezug auf die RD-Kosten ist. Die Modeauswahl als ein Ergebnis der Rate-Distortion-Optimierung muss als Teil des Bitstroms an den Decoder signalisiert werden, da der Decoder mangels des nur am Encoder vorliegenden Originalsignals keine eigene Rate-Distortion-

Optimierung durchführen kann. Es wird ein zusätzliches Bin[6] namens `comic_flag` in die CU-Syntax, §7.3.8.5 des HEVC-Standards, eingeführt. Dieses Bin wird mittels CABAC unter Verwendung eines während der Codierung fortlaufend auf Basis zuvor codierter Syntaxelemente angepassten Kontexts codiert. Die Initialisierung des Kontexts erfolgt auf die gleichen Werte wie für das Syntaxelement `pred_mode_flag`, welches die Entscheidung zwischen Intra- und Intra-Codierung signalisiert. Die Erweiterung der Syntaxtabelle ist in Tabelle 5.3 hervorgehoben.

Tabelle 5.3: Syntaxtabelle für die Integration in einen Videocodec. Ein Bin in der CU-Syntax dient der Signalisierung, ob das vorgeschlagene Verfahren für die Codierung eines Blocks verwendet wird. Die Tabelle basiert auf §7.3.8.5 des HEVC-Standards. Ergänzungen wurden in gelb hinterlegt. $ae(v)$ steht für ein Syntaxelement, welches mit binärer arithmetischer Entropiecodierung unter Verwendung eines während der Codierung fortlaufend auf Basis zuvor codierter Syntaxelemente angepassten Kontexts codiert wird.

| coding_unit( x0, y0, log2CbSize ) { | **Descriptor** |
|---|---|
| if( transquant_bypass_enabled_flag ) | |
| **cu_transquant_bypass_flag** | ae(v) |
| if( slice_type != I ) | |
| **cu_skip_flag**[ x0 ][ y0 ] | ae(v) |
| nCbS = ( 1 << log2CbSize ) | |
| if( cu_skip_flag[ x0 ][ y0 ] ) | |
| prediction_unit( x0, y0, nCbS, nCbS ) | |
| else { | |
| if( slice_type != I ) | |
| **pred_mode_flag** | ae(v) |
| if( CuPredMode[ x0 ][ y0 ] != MODE_INTRA \|\| log2CbSize == MinCbLog2SizeY ) | |
| **part_mode** | ae(v) |
| if( CuPredMode[ x0 ][ y0 ] == MODE_INTRA ) { | |
| **comic_flag**[ x0 ][ y0 ] | ae(v) |
| if ( !comic_flag[ x0 ][ y0 ] ) { | |
| […] | |
| } | |
| } | |
| } else { | |
| […] | |
| } | |
| […] | |
| } | |
| } | |

6 Zum Unterschied zwischen Bins und Bits siehe Fußnote 14 in Kapitel 2.

Die Semantik für das eingeführte Syntaxelement lautet:

> *comic_flag[xo][yo] equal to 1 specifies that the CoMIC mode is used to code the current coding unit. comic_flag[xo][yo] equal to 0 specifies that the coding unit is not coded with the CoMIC mode. The array indices xo, yo specify the location (xo,yo) of the top-left luma sample of the considered coding block relative to the top-left luma sample of the picture. When not present, the value of comic_flag is inferred to be equal to 0.*

Es ist ersichtlich, dass nur geringfügige Änderungen an Syntax und Semantik zur Integration der vorgeschlagenen Verfahren notwendig sind.

Als Metriken werden die Datenrate des sich ergebenen Bitstroms sowie der MSE und hieraus berechnet das PSNR zur Qualitätsbeurteilung verwendet. Aus den Datenraten und den PSNR-Werten lassen sich wie zuvor beschrieben die BD-Raten berechnen. Die Encoderlaufzeit wird ebenfalls gemessen. Für diese ist eine deutliche Steigerung zu erwarten, da im Rahmen der RD-Optimierung die vorgeschlagenen Verfahren für jede Blockgröße getestet werden. Dies impliziert, dass jedes Pixel — wie auch für andere Prädiktionsverfahren — mehrfach durch die vorgeschlagenen Verfahren prädiziert werden muss, weil die Ergebnisse für eine Blockgröße weder für die Prädiktion mit anderen Blockgrößen wiederverwendet werden noch kombiniert werden können. Für die Integration in einen Videocodec wird zusätzlich die Decoderlaufzeit analysiert. Anders als bei der zuvor betrachteten CoMIC-Implementierung ist die Decoderlaufzeit für die HM-Implementierung von Relevanz. Der Unterschied ergibt sich aus der Verfügbarkeit von alternativen Prädiktionsverfahren gemäß des HEVC-Standards, welche auf Blockebene anstatt des vorgeschlagenen Verfahrens verwendet werden können. Da die vorgeschlagenen Verfahren nur dann eingesetzt werden, wenn sie im Rahmen der RD-Optimierung zu den geringsten Kosten führen, hängt die Veränderung der Decoderlaufzeit von der Verwendungshäufigkeit der vorgeschlagenen Verfahren ab. Die Verwendungshäufigkeit wird ebenfalls gemessen.

Im Folgenden wird der Experimentaufbau für die Evaluierung der vorgeschlagenen Verfahren in einem Videocodec ausgeführt. Im Rahmen der Standardisierungsaktivitäten für HEVC wurden Common Test Conditions (CTC) für die Evaluierung von auf HM aufbauenden Codierungsverfahren festgelegt [10]. Die Motivation für die Verwendung der CTC ergibt sich daraus, dass diese präzise definierte Einstellungen des Encoders vorgeben, um so eine Vergleichbarkeit von Ergebnissen zu ermöglichen. Im Gegensatz hierzu setzen kommerzielle Encoder auf eine Vielzahl von Encoder-Optimierungen und Ratenkontroll-Algorithmen.

Da sich hierdurch viele Effekte überlagern, ist es schwierig den Mehrwert von neuen Codierungsverfahren zu messen. Deshalb werden in der Literatur und in den Standardisierungsgremien Encodereinstellungen ohne viele Optimierungen und Ratenkontrolle verwendet [140].

In den CTC sind drei Konfigurationen definiert:

1. *All Intra*: In dieser Konfigurationen werden alle Bilder ohne Referenzierung anderer, zuvor codierter, Bilder, beispielsweise mittels bewegungskompensierender Prädiktion, codiert. Hierdurch sollen die Intra-Prädiktionsverfahren sowie die Prädiktionsfehlercodierung für die durch die Intra-Prädiktion erzeugten Prädiktionsfehler evaluiert werden. Die Einzelbildcodierung kann als Spezialfall für die Anwendung dieser Konfiguration gesehen werden.

2. *Low Delay*: Diese Konfiguration zielt auf Anwendungen die eine geringe Latenz erfordern. Hierzu zählen beispielsweise Videokonferenzen. Das erste Bild einer Sequenz kann ausschließlich intraprädiziert werden. Für die Blöcke in den weiteren Bildern der Sequenz kann bei dieser Konfiguration zusätzlich zur Intra-Prädiktion die Inter-Prädiktion mit Referenz auf zuvor angezeigte Bilder verwendet werden. Da die Bilder in der gleichen Reihenfolge codiert werden, in der sie auch angezeigt werden, ergibt sich keine durch die GOP-Struktur bedingte Latenz.

3. *Random Access*: Mit dieser Konfiguration werden klassische Rundfunk-Anwendungen sowie andere Anwendungen mit der Notwendigkeit eines wahlfreien Zugriffs, beispielsweise Streaming oder Blu-rays, abgebildet. Die Bilder werden einer hierarchischen GOP-Struktur entsprechend codiert. Für die zu codierenden Blöcke können sowohl die Intra-Prädiktion als auch die Inter-Prädiktion eingesetzt werden. Im Gegensatz zur *Low Delay*-Konfiguration können in der *Random Access*-Konfiguration nicht nur vorherige sondern auch zukünftige Bilder referenziert werden. Dieses wird dadurch ermöglicht, dass sich die Codierungsreihenfolge der Bilder von deren Anzeigereihenfolge unterscheidet. Hierdurch entsteht eine zusätzliche Latenz für den Decodierungsprozess. In regelmäßigen Abständen wird ein ausschließlich intra-codiertes Bild verwendet, um den wahlfreien Zugriff zu ermöglichen. Für diesen Abstand wird in Abhängigkeit der Bildwiederholrate der jeweiligen Sequenz dasjenige Vielfache der GOP-Größe von 8 gewählt, dass einem I-Bild-Abstand von einer Sekunde am nächsten kommt.

In den CTC werden ferner 24 Videosequenzen mit unterschiedlichen Auflösungen und Charakteristiken aufgeführt. Diese werden ebenfalls

alle für die vorliegende Arbeit verwendet. Die Quantisierungsparameter werden, ebenfalls den CTC folgend, auf 22, 27, 32, und 37 festgelegt.

Für die Evaluierung der vorgeschlagenen Verfahren werden die CTC in zwei Punkten erweitert, um zusätzliche Experimente durchzuführen.

Zum einen ließe sich gegen die Verwendung der CTC-Sequenzen einwenden, dass diese auch für die Entwicklung des HEVC-Standards verwendet wurden. Deshalb werden zusätzliche Sequenzen codiert und die Ergebnisse getrennt angegeben. Es handelt sich um die Bristol Vision Institute (BVI)-Texturdatenbank [97] und die *Bike*-Datenbank [68]. In der erstgenannten beinhalten die Sequenzen verschiedene Texturen. Die Sequenzen sind wie die CTC-Sequenzen unkomprimiert. Bei den Sequenzen aus der zweiten zusätzlichen Datenbank handelt es sich um sogenannten *User-generated Content*, der anders als die CTC- und BVI-Sequenzen nicht mit höchstqualitativen Kameras aufgenommen wurde. Die Sequenzen wurden bereits in der Kamera codiert. Es handelt sich also bei den durchgeführten Experimenten für diese Sequenzen um eine Transcodierung. Dieses entspricht der typischen Anwendung für diese Art von Inhalten. Bei beidem handelt es sich um wichtige Charakteristiken, die in den CTC-Sequenzen unterrepräsentiert sind. Die zusätzlichen Ergebnisse werden getrennt präsentiert.

Zum anderen wurde in der eigenen Vorarbeit [70] erkannt, dass die Verwendung der vier in den CTC definierten Quantisierungsparameter zu teils unrealistisch hohen Qualitäten und Datenraten führt. So werden 4K-Sequenzen mit teils über 200 MBit/s codiert, während für die gleiche Auflösung in realen Streaming-Anwendungen teils nur etwas mehr als ein Zehntel hiervon verwendet wird [89]. Auch die Qualität des rekonstruierten Videos ist mit teils über 45dB für mit einer Kamera aufgenommene Videosequenzen und über 50dB für computergenerierte Sequenzen sehr hoch. Narroschke gibt für Rundfunkqualität einen Wert von 36dB an [88]. Deshalb wird ein zusätzlicher Satz an Quantisierungsparametern, 32, 37, 42, 47, verwendet. Die Verwendung dieser Quantisierungsparameter führt zu niedrigeren Qualitäten und Datenraten, wie sie insbesondere für Streaming-Anwendungen über Mobilfunknetze, Echtzeitanwendungen wie Videokonferenzen oder noch häufiger bei der Videotelefonie mit mobilen Endgeräten zu beobachten sind. Auch hier werden die Ergebnisse separat angegeben.

Zur Verringerung des Rechenaufwands wurden von jeder Sequenz die ersten 64 Bilder codiert. Unter Berücksichtigung der Tatsache, dass in dieser Arbeit Intra-Codierungsverfahren vorgeschlagen werden, erscheint diese Einschränkung vertretbar. Für Intra-Codierungsverfahren erscheint durch die Codierung von 64 Bildern von einer hohen Anzahl an unterschiedlichen Testsequenzen der zu erwartende Erkenntnisgewinn

höher zu sein als durch die Codierung von jeweils mehr Bildern von weniger Sequenzen. Diese Annahme wird gestärkt durch die Beobachtung, dass lediglich eine der über fünfzig verwendeten Testsequenzen (*Kimono*) einen Szenenwechsel enthält.

Die verwendeten Testsequenzen werden in Abbildung 5.13 veranschaulicht.

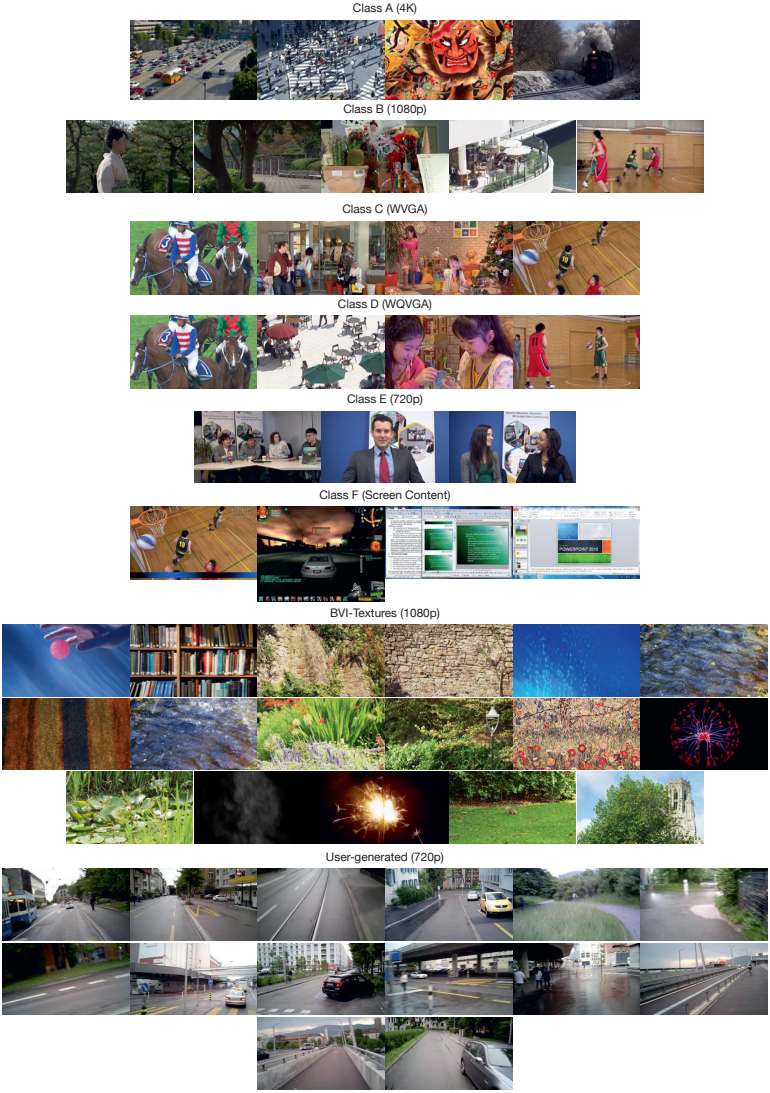



Abbildung 5.13: Veranschaulichung der verwendeten Testsequenzen

Für die Messung der Codierungseffizienz werden die BD-Raten [7, 8] mit der Implementierung aus [10] berechnet. Für die drei Farbraumkomponenten sind jeweils PSNR-Werte messbar — pro QP einer. Die jeweiligen Werte werden mit $\mathbf{PSNR_Y}$, $\mathbf{PSNR_{Cb}}$ und $\mathbf{PSNR_{Cr}}$ bezeichnet. Bei der Videocodierung entsteht ein Bitstrom, der alle drei Farbraumkomponenten gemeinsam repräsentiert. Hierdurch ist es schwierig, einzelne Bitanzahlen den jeweiligen Komponenten zuzuordnen. Typischerweise verursacht die Luma-Komponente einen höheren Anteil der Gesamtdatenrate als die Chroma-Komponenten. Deshalb werden die Ergebnisse als gewichtete Überlagerung der Ergebnisse der einzelnen Farbraumkomponenten berechnet. Als Gewichtungsfaktoren werden für 4:2:0-Farbuntertastungen in der Regel die Werte 6/1/1 für die Komponenten Y/Cb/Cr und die Faktoren 4/1/1 für 4:4:4-Farbunterabtastungen verwendet.

Für die Berechnung der gewichteten Überlagerung stehen zwei Vorgehensweisen zur Auswahl: Entweder könnten zunächst drei BD-Raten für die einzelnen Komponenten ($\mathrm{BD_Y}$, $\mathrm{BD_{Cb}}$, $\mathrm{BD_{Cr}}$) berechnet und anschließend durch gewichtete Überlagerung die BD-Rate für die drei Farbraumkomponenten gemeinsam ($\mathrm{BD_{YCbCr}}$) ermittelt werden:

$$\mathrm{BD_Y} = \mathcal{BD}\left(\mathbf{r}, \mathbf{PSNR_Y}\right), \tag{5.3}$$

$$\mathrm{BD_{Cb}} = \mathcal{BD}\left(\mathbf{r}, \mathbf{PSNR_{Cb}}\right), \tag{5.4}$$

$$\mathrm{BD_{Cr}} = \mathcal{BD}\left(\mathbf{r}, \mathbf{PSNR_{Cr}}\right), \tag{5.5}$$

$$\mathrm{BD_{YCbCr}} = \frac{6 \times \mathrm{BD_Y} + \mathrm{BD_{Cb}} + \mathrm{BD_{Cr}}}{8}. \tag{5.6}$$

Hierbei steht $\mathcal{BD}(\mathbf{r}, \mathbf{PSNR_{Y/Cb/Cr}})$ für die Berechnung der BD-Rate in Abhängigkeit der als Argument angegebenen Gesamt-Datenraten $\mathbf{r}$ sowie der zu der entsprechenden Farbraumkomponente gehörenden PSNR-Werte. Alternativ könnten zunächst die PSNR-Werte überlagert werden, um anschließend eine BD-Rate für die Gesamt-Datenrate und den gewichteten PSNR-Mittelwert zu berechnen:

$$\mathbf{PSNR_{YCbCr}} = \frac{6 \times \mathbf{PSNR_Y} + \mathbf{PSNR_{Cb}} + \mathbf{PSNR_{Cr}}}{8}, \tag{5.7}$$

$$\mathrm{BD_{YCbCr}} = \mathcal{BD}\left(\mathbf{r}, \mathbf{PSNR_{YCbCr}}\right). \tag{5.8}$$

In dem internationalen Standardisierungsgremium JVET von ISO, IEC und ITU-T für Videocodierung wird die Auffassung vertreten, dass die Variante aus Gleichung 5.8 vorzuziehen sei, da bei der Variante aus Gleichung 5.6 Ausreißer in den Chroma-Komponenten einen zu starken Einfluss hätten [114]. Deshalb wird für diese Arbeit die Variante aus Gleichung 5.8 verwendet.

Die folgenden Kombinationen an Codierungsverfahren werden untersucht:

1. Polynomielles Konturmodell +
   Abtastwertprädiktion mit neuronalen Netzwerken

2. Stochastisches Konturmodell +
   Abtastwertprädiktion mit neuronalen Netzwerken

Die Beschränkung auf diese beiden Kombinationen wird damit begründet, dass bereits in dem Experiment mit dem CoMIC-Codec gezeigt wurde, dass die Abtastwertprädiktion mit neuronalen Netzwerk überlegen mit Bezug auf die *Along-contour*-Abtastwertprädiktion ist und der Rechenaufwand für die Durchführung der Simulationen somit begrenzt werden kann.

### 5.2.1 *Polynomielles Konturmodell und Abtastwertprädiktion mit neuronalen Netzwerken*

Zunächst wird in diesem Abschnitt die Kombination aus dem polynomiellen Konturmodell aus der Vorarbeit [69] und dem in dieser Arbeit vorgeschlagenen Verfahren zur Abtastwertprädiktion mittels neuronaler Netzwerke evaluiert. Eine tabellarische Übersicht der BD-Raten für alle Konfigurationen befindet sich in Tabelle 5.4.

Es ist ersichtlich, dass für fast alle Sequenzen eine Verbesserung der Codierungseffizienz gemessen wird. Die Tatsache, dass trotz der Rate-Distortion-Optimierung für einige wenige Sequenzen Verluste gemessen werden können, lässt sich damit erklären, dass in der gewählten Implementierung die Entscheidung für das Codierungsverfahren basierend auf der Luma-Komponente getroffen werden muss, bevor die Codierung der Chroma-Komponenten beginnt. In einzelnen Fällen kann es dazu kommen, dass die Codierung für eine Chroma-Komponente nicht so gut funktioniert, obwohl für die Luma-Komponente ein gutes Ergebnis erzielt wurde und der CoMIC-Mode deshalb ausgewählt wurde. Da die Datenraten für die drei Komponenten nicht getrennt betrachtet werden können, hat dieses einen nicht vermeidbaren Einfluss auf die BD-Raten. Hierzu lässt sich die Hypothese aufstellen, dass die Codierungsgewinne durch eine ausgefeiltere RD-Optimierung höchstwahrscheinlich weiter gesteigert werden könnten.

Für *All Intra* werden BD-Raten von bis zu $-3.44\%$ gemessen. Interessante Erkenntnisse lassen sich durch die Analyse, für welche Arten von Sequenzen die größten Gewinne erzielt werden, gewinnen. Hierfür werden die Mittelwerte im unteren Bereich der Tabelle 5.4 betrachtet. Aus

den Daten geht hervor, dass die größten Gewinne für Sequenzen mit sehr großen Auflösungen (Class A, 4K) und mit *User-generated Content* erzielt werden. Die geringsten Gewinne beziehungsweise teilweise sogar Verluste werden für Sequenzen mit sehr geringen Auflösungen (Class D, $416 \times 240$) und computergenerierten Inhalten (Class F, *Screen Content*) gemessen.

Die für die Prädiktion verwendeten neuronalen Netzwerke wurden mit Trainingsdaten trainiert, in denen Auflösungen von $3008 \times 2000$, $4288 \times 2848$ und $4928 \times 3264$ vorkommen. Diese Auflösungen sind in der gleichen Größenordnung wie die Auflösung der 4K Sequenzen, was nahe legt, dass die Signalcharakteristiken ähnlicher als bei den sehr niedrigen Auflösungen sind. Die *User-generated Content*-Sequenzen wurden mit einer nicht-professionellen Kamera aufgenommen. Verglichen mit den mit professionellen Kameras aufgenommenen CTC-Sequenzen haben diese Sequenzen eine geringere Ausgangsqualität, beispielsweise in Bezug auf das Vorhandensein hoher Frequenzen. Dieses begünstigt die Prädiktion mit den per SATD trainierten Netzwerken ebenfalls. Da computergenerierte Inhalte in den Trainingsdaten der Netzwerke nicht vorkommen, ist es wenig verwunderlich, dass das vorgeschlagene Codierungsverfahren hierfür nicht so effizient funktioniert.

Des Weiteren ist festzustellen, dass die Codierungsgewinne im Mittel für die Konfigurationen *Random Access* und *Low Delay* in der gleichen Größenordnung wie die Gewinne für *All Intra* sind. Dieses zeigt, dass Intra-Codierungsverfahren auch für Bilder, für die eine bewegungskompensierende Prädiktion verwendet werden kann, von großer Bedeutung sind. Auch in diesen Bildern wird ein Anteil der Blöcke intra-codiert, beispielsweise neu im Bild erscheinende Inhalte. Für die Interpretation der Ergebnisse sind zwei Aspekte zu berücksichtigen: Zum einen werden weniger Blöcke intra-codiert als in der *All Intra*-Konfiguration[7]. Zum anderen ist die Gesamtdatenrate niedriger. Wenn nun davon ausgegangen wird, dass die absoluten Datenrateneinsparungen für die intra-codierten Blöcke vergleichbar sind und diese für die BD-Raten-Berechnung mit der Gesamtdatenrate in Bezug gesetzt werden, dann lässt sich erklären, warum auch für die *Random Access*- und *Low Delay*-Konfiguration nennenswerte Codierungsgewinne erzielt werden.

7 Ein entsprechender Nachweis wird im weiteren Verlauf des Kapitels bei der Analyse der Verwendungshäufigkeit erbracht.

Tabelle 5.4: BD-Raten für die HM-Implementierung des polynomiellen Konturmodells in Kombination mit neuronalen Netzwerken zur Abtastwertprädiktion. HR = Hohe Datenraten, NR = Niedrige Datenraten. Negative BD-Raten zeigen eine gesteigerte Codierungseffizienz an.

| | | All Intra | | Random Access | | Low Delay | |
|---|---|---|---|---|---|---|---|
| **Kategorie** | **Sequenz** | **HR** | **NR** | **HR** | **NR** | **HR** | **NR** |
| Class A (4K) | Traffic | -0.36% | -0.87% | -0.33% | -0.76% | -0.12% | -0.52% |
| | People on Street | -0.62% | -1.05% | -0.53% | -1.42% | -0.37% | -1.18% |
| | Nebuta | -0.18% | -0.41% | -0.02% | -0.04% | 0.01% | -0.37% |
| | Steam Locomotive | -0.49% | -3.44% | -0.53% | -1.43% | -0.27% | -1.01% |
| Class B (1080p) | Kimono | -0.83% | -0.71% | -0.23% | -0.39% | -0.27% | -0.49% |
| | Park Scene | -0.69% | -1.65% | -0.40% | -0.92% | -0.22% | -0.53% |
| | Cactus | -0.22% | -0.71% | -0.27% | -0.59% | -0.26% | -0.58% |
| | BQ Terrace | -0.08% | -0.35% | -0.10% | -0.05% | -0.11% | 0.09% |
| | Basketball Drive | -0.17% | -1.04% | -0.49% | -0.76% | -0.35% | -0.89% |
| Class C (WVGA) | Race Horses | -0.10% | -0.76% | -0.37% | -0.71% | -0.26% | -0.84% |
| | BQ Mall | 0.08% | -1.07% | -0.02% | -0.54% | 0.05% | -0.20% |
| | Party Scene | 0.06% | 0.33% | -0.13% | -0.84% | 0.07% | 0.14% |
| | Basketball Drill | -0.14% | 0.03% | -0.33% | -0.85% | -0.17% | 0.74% |
| Class D (WQVGA) | Race Horses | 0.43% | 0.46% | -0.20% | -0.27% | 0.22% | 0.74% |
| | BQ Square | -0.09% | -0.40% | -0.06% | -0.15% | -0.18% | -0.08% |
| | Blowing Bubbles | 0.00% | -0.63% | -0.16% | -0.18% | -0.05% | -0.05% |
| | Basketball Pass | 0.03% | 0.65% | -0.18% | -0.14% | 0.04% | -0.36% |
| Class E (720p) | Four People | -0.36% | -0.10% | -0.13% | -0.42% | -0.40% | -0.60% |
| | Johnny | 0.14% | -0.70% | -0.58% | -0.18% | -0.63% | -0.65% |
| | Kristen and Sara | -0.44% | -0.77% | 0.14% | -0.01% | -0.10% | -0.55% |
| Class F (Screen Content) | Basketball Drill Text | 0.08% | 0.03% | -0.25% | -0.45% | 0.09% | 0.22% |
| | China Speed | -0.17% | -0.25% | -0.14% | 0.01% | -0.02% | -0.35% |
| | Slide Editing | 0.14% | 0.25% | 0.24% | -0.21% | 0.20% | 0.11% |
| | Slide Show | -0.12% | 0.90% | 0.07% | 0.14% | -0.07% | 0.52% |
| BVI-Textures (1080p) | Ball Under Water | -1.00% | -1.87% | -1.68% | -3.17% | -1.42% | -3.29% |
| | Bookcase | -0.55% | -1.33% | -0.19% | -0.56% | -0.08% | 0.60% |
| | Bricks and Bushes | -0.39% | -0.81% | -0.21% | -0.59% | -0.03% | -0.43% |
| | Bricks Leaves | -0.47% | -1.42% | -0.62% | -1.35% | -0.68% | -1.49% |
| | Bubbles Clear | -0.03% | -0.04% | -1.84% | -1.85% | -1.78% | -1.78% |
| | Calming Water | -0.64% | -0.48% | -0.84% | -1.23% | -0.67% | -0.83% |
| | Carpet | -0.28% | -0.44% | -0.20% | -0.15% | -0.43% | 0.23% |
| | Drops on Water | -0.77% | -1.07% | -0.78% | -1.03% | -0.66% | -0.65% |
| | Flowers 2 | -0.59% | -0.75% | -0.19% | -0.41% | -0.24% | -0.22% |
| | Lamp Leaves | -0.55% | -1.01% | -0.23% | -0.63% | -0.14% | -0.41% |
| | Paiting Tilting | -0.31% | -0.73% | -0.30% | -0.57% | -0.22% | -0.34% |
| | Plasma Free | 0.06% | -0.20% | -0.77% | -1.30% | -0.47% | -0.64% |
| | Pond Dragonflies | -0.34% | -0.61% | -0.28% | -0.70% | -0.09% | -0.33% |
| | Smoke Clear | -0.15% | 0.48% | -0.38% | -1.76% | -1.29% | -4.54% |
| | Sparkler | -0.36% | -1.61% | -1.79% | -1.87% | -1.56% | -1.67% |
| | Squirrel | -0.35% | -0.83% | -0.73% | -1.02% | -0.37% | -0.68% |
| | Tree Willis | -0.28% | -0.63% | -0.08% | -0.45% | -0.12% | -0.35% |
| User-generated (720p) | Bike 1 | -0.62% | -1.99% | -0.41% | -1.02% | 0.12% | -0.07% |
| | Bike 2 | -0.92% | -0.64% | -0.21% | -1.35% | -0.20% | -0.82% |
| | Bike 3 | 0.07% | -2.04% | -0.29% | -0.80% | -0.38% | -0.22% |
| | Bike 4 | -0.38% | -0.12% | -0.57% | -1.29% | -0.54% | -1.36% |
| | Bike 5 | -1.24% | -0.86% | -0.19% | -1.40% | -0.32% | -1.35% |
| | Bike 6 | -0.57% | -1.44% | -0.38% | -0.50% | -0.48% | -0.94% |
| | Bike 7 | -0.85% | -0.91% | -0.46% | -0.05% | -0.56% | -0.81% |
| | Bike 8 | -0.17% | 0.25% | -0.33% | -0.84% | -0.51% | -0.25% |
| | Bike 9 | -0.62% | -1.24% | -0.37% | -1.25% | -0.28% | -0.88% |
| | Bike 10 | -0.33% | -1.37% | -0.52% | -0.74% | -0.41% | -0.46% |
| | Bike 11 | -0.65% | -1.59% | -0.48% | -1.52% | -0.37% | -0.80% |
| | Bike 12 | 0.15% | 0.43% | -0.12% | -0.42% | -0.09% | -0.02% |
| | Bike 13 | -1.02% | -0.50% | -0.29% | -0.67% | -0.01% | -0.40% |
| | Bike 14 | -0.37% | -2.30% | -2.43% | -5.75% | -1.44% | -4.51% |
| | Mittelwert | -0.34% | -0.73% | -0.42% | -0.86% | -0.34% | -0.66% |
| | | | | | | | |
| | Class A (4K) | -0.41% | -1.44% | -0.35% | -0.91% | -0.19% | -0.77% |
| | Class B (1080p) | -0.40% | -0.90% | -0.30% | -0.54% | -0.24% | -0.48% |
| | Class C (WVGA) | -0.03% | -0.37% | -0.21% | -0.73% | -0.08% | -0.04% |
| | Class D (WQVGA) | 0.09% | 0.02% | -0.15% | -0.18% | 0.01% | 0.06% |
| | Class E (720p) | -0.22% | -0.52% | -0.19% | -0.20% | -0.38% | -0.60% |
| | Class F (Screen Content) | -0.02% | 0.23% | -0.02% | -0.13% | 0.05% | 0.12% |
| | BVI-Textures (1080p) | -0.41% | -0.78% | -0.65% | -1.10% | -0.60% | -0.99% |
| | User-generated (720p) | -0.54% | -1.02% | -0.50% | -1.26% | -0.39% | -0.92% |

### 5.2.2 *Stochastisches Konturmodell und Abtastwertprädiktion mit neuronalen Netzwerken*

In diesem Abschnitt wird die Kombination aus den beiden in dieser Arbeit vorgeschlagenen Verfahren evaluiert. Hierbei wird untersucht, ob es durch das stochastische Konturmodell nicht nur für die *Along contour*-Abtastwertprädiktion, wie bereits in Abschnitt 5.1.1 gezeigt, einen Mehrwert gegenüber dem polynomiellen Konturmodell bietet, sondern auch wenn es mit der vorgeschlagenen Abtastwertprädiktion mit neuronalen Netzwerken kombiniert wird. Es werden die identischen Encoderkonfigurationen und Testsequenzen verwendet wie im vorherigen Experiment. Eine tabellarische Übersicht der BD-Raten für alle Konfigurationen befindet sich in Tabelle 5.5.

Es ist ersichtlich, dass die vorgeschlagenen Verfahren für die Inhalte, für die bereits die Kombination aus dem polynomiellen Konturmodell mit der Abtastwertprädiktion basierend auf neuronalen Netzwerken zu nennenswerten Codierungsgewinnen führten, ebenfalls gut funktionieren. Hierbei handelt es sich um die Kategorien mit hochauflösenden Sequenzen (Class A mit 4K-Sequenzen und Class B 1080p-Sequenzen) sowie für *User-generated Content*. Darüber hinaus ist zu beobachten, dass für die schwierig zu codierenden Texturen in der BVI-Datenbank der Codierungsgewinn deutlich gesteigert werden kann. Im Mittel betragen die gemessenen BD-Raten für die AI-Konfiguration $-0,54\%$ (für die hohen Datenraten) und $-1,0\%$ für die niedrigen Datenraten. Der gemessene Maximalwert für die Steigerung der Codierungseffizienz ist mit einer BD-Rate von $-5\%$ in etwa genauso groß wie für das Experiment mit dem polynomiellen Konturmodell.

Um den Mehrgewinn durch das stochastische Konturmodell genauer zu quantifizieren wird die Differenz der jeweils gemessenen BD-Raten gebildet:

$$\text{Mehrgewinn} = \text{BD-Rate}_{\text{Stochastisch}} - \text{BD-Rate}_{\text{Polynomiell}}$$

Negative Zahlenwerte sagen hierbei aus, um wieviel zusätzliche Prozentpunkte die Codierungseffizienz durch das vorgeschlagene Verfahren im Vergleich zu der eigenen Vorarbeit gesteigert werden konnte. Die Ergebnisse sind in Tabelle 5.6 zusammengefasst. Aus der Tabelle lässt sich ablesen, dass die Verwendung des stochastischen Konturmodells einen Mehrwert für die Codierungseffizienz bietet. Dieses zeigt sich sowohl in den gemittelten BD-Raten als auch in der Anzahl an Sequenzen, für welche die Codierungseffizienz zusätzlich gesteigert werden kann. Für die *All Intra*-Konfiguration wird die Codierungseffizienz im Mittel um weitere $0,21$ Prozentpunkte (für hohe Datenraten) beziehungsweise $0,27$

Prozentpunkte (für niedrige Datenraten) gesteigert. Dieses entspricht bezogen auf die die ursprünglichen BD-Raten prozentualen Steigerungen von 58% und 36%. Für einige Sequenzen werden Steigerungen um mehr als einen Prozentpunkt gemessen.

Wie auch für das vorhergehende Experiment zeigt sich, dass auch für die Encoderkonfigurationen mit der Möglichkeit zur Verwendung der bewegungskompensierenden Prädiktion die Codierungseffizienz gesteigert werden kann. Für *Random Access* werden BD-Raten von $-0,56\%$ (für hohe Datenraten) und $-1,05\%$ (für niedrige Datenraten) gemessen. Für *Low Delay* liegen die entsprechenden Werte bei $-0,42\%$ und $-0,85\%$. Die Erklärung für diese Codierungsgewinne liegt wie auch zuvor in der Annahme, dass auch in diesen Konfigurationen ein Teil der Pixel intra-prädiziert wird. Zwar ist davon auszugehen, dass weniger Pixel intra-prädiziert werden — eine Annahme, die nachfolgend noch näher untersucht wird — jedoch die für diese Pixel eingesparte absolute Datenrate zu einer ebenfalls niedrigeren Gesamtdatenrate in Bezug gesetzt wird und somit die Codierungsgewinne in ähnlicher Höhe wie für die *All Intra*-Konfiguration erklärt werden können.

Zur weiteren Analyse wird ermittelt, welcher Anteil der codierten Pixel durch das CoMIC-Verfahren prädiziert wird. Hierbei wird bewusst der Anteil an Pixeln und nicht der Anteil an Blöcken betrachtet. Dieses ist notwendig, da die Blöcke unterschiedlich groß sein können und somit der Anteil an mit dem CoMIC-Verfahren codierten Blöcken nur eine geringe Aussagekraft hätte. Die Ergebnisse sind in Tabelle 5.7 zusammengefasst. Im Mittel werden die vorgeschlagenen Verfahren für die *All Intra*-Konfiguration für $12,9\%$ (hohe Datenraten) beziehungsweise für $17,85\%$ (niedrige Datenraten) der codierten Pixel verwendet. Für die Konfiguration *Random Access* liegen die Werte bei $3,95\%$ und $3,53\%$, für die *Low Delay*-Konfiguration bei $3,8\%$ und $3,7\%$. Somit wird das CoMIC-Verfahren regelmäßig durch die RD-Optimierung als bestes Codierungsverfahren ausgewählt. Ferner lässt sich festhalten, dass die Verwendungshäufigkeit erwartungsgemäß für die Encoderkonfigurationen, welche eine bewegungskompensierende Prädiktion zulassen, geringer ist, aber dennoch für ein Intra-Prädiktionsverfahren häufig ist.

Die Steigerung der Komplexität wird anhand der Encoder- und Decoderlaufzeiten analysiert. Die Ergebnisse sind in Tabelle 5.8 zusammengefasst. Zur Einordnung ist anzumerken, dass die Laufzeiten aufgrund der sehr großen Anzahl an Simulationen mittels einer stark heterogenen Mischung aus verschiedenen Computern gemessen wurden. Die verwendeten CPU-Typen sind: Intel Xeon CPU E5-2680 v3 (2.50GHz), Intel Xeon CPU E5-2670 (2.60GHz), Intel Xeon Gold 5120 CPU (2.20GHz), Intel Xeon CPU E5-2690 v2 (3.00GHz), Intel Core i9-9900K CPU (3.60GHz) und Intel

Core i7-3770K CPU (3.50GHz). Wie bereits aufgrund der Analyse der Komplexität im Bildcodec erwartet wurde, ist die Steigerung der Komplexität sehr groß. Hierdurch sind die Auswirkungen durch die unterschiedlichen CPU-Typen im Rahmen einer akzeptablen Messungenauigkeit.

Abschließend ist erwähnenswert, dass die Gewichte der neuronalen Netzwerke sowie alle weiteren Parameter des Codierungsverfahrens für alle codierten Sequenzen gleich waren. Durch eine Adaptivität, beispielsweise durch das Vorhalten mehrere Netzwerkmodelle für unterschiedliche Signalcharakteristiken, ließe sich die Codierungseffizienz höchstwahrscheinlich weiter steigern.

Tabelle 5.5: BD-Raten für die HM-Implementierung des stochastischen Konturmodells in Kombination mit neuronalen Netzwerken zur Abtastwertprädiktion. HR = Hohe Datenraten, NR = Niedrige Datenraten. Negative BD-Raten zeigen eine gesteigerte Codierungseffizienz an.

| | | All Intra | | Random Access | | Low Delay | |
|---|---|---|---|---|---|---|---|
| **Kategorie** | **Sequenz** | **HR** | **NR** | **HR** | **NR** | **HR** | **NR** |
| Class A (4K) | Traffic | -0.51% | -1.12% | -0.43% | -0.94% | -0.17% | -0.68% |
| | People on Street | -0.87% | -1.43% | -0.79% | -1.82% | -0.47% | -1.45% |
| | Nebuta | -0.53% | -0.91% | -0.17% | -0.28% | 0.03% | -0.16% |
| | Steam Locomotive | -0.63% | -3.38% | -0.06% | -0.89% | -0.35% | -0.70% |
| Class B (1080p) | Kimono | -1.25% | -1.23% | -0.46% | -0.80% | -0.25% | -0.81% |
| | Park Scene | -0.84% | -2.00% | -0.55% | -1.28% | -0.33% | -1.06% |
| | Cactus | -0.30% | -1.02% | -0.37% | -0.88% | -0.36% | -0.99% |
| | BQ Terrace | -0.14% | -0.51% | -0.09% | 0.48% | -0.16% | -0.18% |
| | Basketball Drive | -0.48% | -2.06% | -0.84% | -1.52% | -0.49% | -1.45% |
| Class C (WVGA) | Race Horses | -0.32% | -0.40% | -0.34% | -0.66% | -0.25% | -1.09% |
| | BQ Mall | 0.13% | -0.58% | -0.32% | -1.28% | 0.04% | -0.51% |
| | Party Scene | -0.08% | -0.41% | -0.18% | -0.89% | -0.06% | -0.31% |
| | Basketball Drill | -0.04% | -0.98% | -0.31% | -0.89% | 0.00% | 0.11% |
| Class D (WQVGA) | Race Horses | 0.26% | -0.16% | -0.22% | -0.97% | 0.06% | 0.06% |
| | BQ Square | 0.04% | -0.48% | 0.35% | 0.43% | 0.00% | -0.40% |
| | Blowing Bubbles | 0.12% | -1.05% | -0.05% | -0.33% | 0.01% | -0.24% |
| | Basketball Pass | -0.03% | 0.67% | -0.06% | -0.13% | 0.31% | -0.65% |
| Class E (720p) | Four People | -0.25% | -0.02% | -0.36% | -0.46% | -0.36% | -0.10% |
| | Johnny | -0.67% | -0.82% | -0.81% | -0.38% | -0.70% | -0.77% |
| | Kristen and Sara | -0.67% | -1.70% | -0.31% | -0.33% | 0.02% | -0.74% |
| Class F (Screen Content) | Basketball Drill Text | -0.24% | -0.13% | -0.23% | -0.34% | 0.04% | 0.09% |
| | China Speed | -0.02% | -0.42% | -0.14% | -0.21% | -0.07% | -0.52% |
| | Slide Editing | -0.04% | -0.46% | -0.22% | -0.21% | 0.12% | 0.08% |
| | Slide Show | -0.08% | -0.45% | -0.28% | 0.32% | 0.13% | 0.56% |
| BVI-Textures (1080p) | Ball Under Water | -1.42% | -1.33% | -2.33% | -4.00% | -1.86% | -3.61% |
| | Bookcase | -0.69% | -2.35% | -0.10% | -0.54% | -0.27% | -0.62% |
| | Bricks and Bushes | -0.68% | -1.16% | -0.27% | -0.81% | -0.08% | -0.40% |
| | Bricks Leaves | -0.67% | -1.39% | -0.81% | -1.57% | -0.82% | -1.44% |
| | Bubbles Clear | 0.00% | -0.68% | -2.41% | -2.65% | -2.39% | -2.63% |
| | Calming Water | -1.06% | -1.52% | -1.52% | -2.35% | -1.30% | -1.74% |
| | Carpet | -0.86% | -1.43% | -0.79% | -1.32% | -0.82% | -0.95% |
| | Drops on Water | -1.06% | -1.62% | -1.32% | -1.93% | -1.16% | -1.12% |
| | Flowers 2 | -1.19% | -1.97% | -0.33% | -1.23% | -0.29% | -0.73% |
| | Lamp Leaves | -0.69% | -1.22% | -0.37% | -0.90% | -0.23% | -0.53% |
| | Painting Tilting | -0.61% | -1.18% | -0.61% | -1.14% | -0.50% | -0.83% |
| | Plasma Free | -0.47% | -0.96% | -1.78% | -2.72% | -1.14% | -1.82% |
| | Pond Dragonflies | -0.64% | -1.36% | -0.37% | -0.87% | -0.15% | -0.71% |
| | Smoke Clear | -0.65% | 1.52% | -1.03% | -2.45% | -1.60% | -4.05% |
| | Sparkler | -0.53% | -1.36% | -2.27% | -3.14% | -1.92% | -2.70% |
| | Squirrel | -0.63% | -1.49% | -0.85% | -1.47% | -0.62% | -1.17% |
| | Tree Willis | -0.47% | -0.91% | -0.16% | -0.49% | -0.16% | -0.40% |
| User-generated (720p) | Bike 1 | -0.99% | -2.10% | -0.34% | -0.86% | -0.23% | -0.03% |
| | Bike 2 | -0.99% | -0.82% | -0.22% | -0.79% | 0.00% | -0.13% |
| | Bike 3 | -0.23% | -0.66% | -0.47% | -0.40% | -0.04% | 0.03% |
| | Bike 4 | -0.53% | -0.88% | -0.49% | -1.36% | -0.62% | -1.49% |
| | Bike 5 | -1.68% | -1.16% | -0.08% | -1.15% | -0.30% | -0.95% |
| | Bike 6 | -0.82% | -1.44% | -0.41% | -0.31% | -0.50% | -0.47% |
| | Bike 7 | -0.61% | -0.50% | -0.66% | -0.76% | -0.57% | -0.93% |
| | Bike 8 | -0.22% | 0.32% | -0.37% | -0.57% | -0.31% | 0.12% |
| | Bike 9 | -0.68% | -1.06% | -0.43% | -1.27% | -0.49% | -0.95% |
| | Bike 10 | -0.64% | -0.82% | -0.34% | -0.04% | -0.36% | -0.32% |
| | Bike 11 | -0.79% | -1.74% | -0.18% | -1.15% | -0.18% | -0.06% |
| | Bike 12 | -0.36% | 0.50% | -0.01% | -0.16% | 0.07% | -0.03% |
| | Bike 13 | -1.16% | -0.38% | -0.26% | -0.24% | 0.04% | -0.28% |
| | Bike 14 | -0.52% | -2.66% | -2.01% | -5.00% | -0.86% | -4.10% |
| | Mittelwert | -0.54% | -1.00% | -0.56% | -1.05% | -0.42% | -0.85% |
| | | | | | | | |
| | Class A (4K) | -0.63% | -1.71% | -0.36% | -0.98% | -0.24% | -0.75% |
| | Class B (1080p) | -0.60% | -1.36% | -0.46% | -0.80% | -0.32% | -0.89% |
| | Class C (WVGA) | -0.08% | -0.59% | -0.29% | -0.93% | -0.07% | -0.45% |
| | Class D (WQVGA) | 0.10% | -0.25% | 0.01% | -0.25% | 0.09% | -0.31% |
| | Class E (720p) | -0.53% | -0.85% | -0.43% | -0.39% | -0.35% | -0.54% |
| | Class F (Screen Content) | -0.09% | -0.37% | -0.22% | -0.11% | 0.06% | 0.05% |
| | BVI-Textures (1080p) | -0.72% | -1.20% | -1.02% | -1.74% | -0.90% | -1.50% |
| | User-generated (720p) | -0.73% | -0.96% | -0.45% | -1.00% | -0.31% | -0.69% |

Tabelle 5.6: Mehrgewinn durch das stochastische Konturmodell im Vergleich mit dem polynomiellen Konturmodell, jeweils in Kombination mit neuronalen Netzwerken zur Abtastwertprädiktion. Es werden die Änderungen der BD-Raten in Prozentpunkten angegeben. Negative Werte zeigen eine gesteigerte Codierungseffizienz an. HR = Hohe Datenraten, NR = Niedrige Datenraten.

| | | All Intra | | Random Access | | Low Delay | |
|---|---|---|---|---|---|---|---|
| **Kategorie** | **Sequenz** | **HR** | **NR** | **HR** | **NR** | **HR** | **NR** |
| Class A (4K) | Traffic | -0.15% | -0.25% | -0.10% | -0.19% | -0.04% | -0.16% |
| | People on Street | -0.25% | -0.38% | -0.25% | -0.40% | -0.11% | -0.27% |
| | Nebuta | -0.35% | -0.51% | -0.16% | -0.24% | 0.02% | 0.21% |
| | Steam Locomotive | -0.14% | 0.06% | 0.47% | 0.55% | -0.08% | 0.31% |
| Class B (1080p) | Kimono | -0.42% | -0.52% | -0.23% | -0.41% | 0.02% | -0.31% |
| | Park Scene | -0.15% | -0.34% | -0.15% | -0.36% | -0.11% | -0.52% |
| | Cactus | -0.08% | -0.30% | -0.10% | -0.28% | -0.10% | -0.41% |
| | BQ Terrace | -0.06% | -0.16% | 0.01% | 0.54% | -0.05% | -0.27% |
| | Basketball Drive | -0.31% | -1.02% | -0.36% | -0.76% | -0.14% | -0.56% |
| Class C (WVGA) | Race Horses | -0.22% | 0.36% | 0.03% | 0.05% | 0.01% | -0.25% |
| | BQ Mall | 0.05% | 0.49% | -0.29% | -0.74% | -0.01% | -0.31% |
| | Party Scene | -0.14% | -0.73% | -0.05% | -0.05% | -0.12% | -0.45% |
| | Basketball Drill | 0.10% | -1.01% | 0.02% | -0.04% | 0.17% | -0.64% |
| Class D (WQVGA) | Race Horses | -0.17% | -0.62% | -0.01% | -0.71% | -0.16% | -0.68% |
| | BQ Square | 0.13% | -0.07% | 0.41% | 0.58% | 0.17% | -0.32% |
| | Blowing Bubbles | 0.12% | -0.42% | 0.11% | -0.15% | 0.06% | -0.19% |
| | Basketball Pass | -0.06% | 0.02% | 0.12% | 0.00% | 0.26% | -0.29% |
| Class E (720p) | Four People | 0.11% | 0.08% | -0.23% | -0.04% | 0.04% | 0.49% |
| | Johnny | -0.81% | -0.13% | -0.23% | -0.20% | -0.07% | -0.12% |
| | Kristen and Sara | -0.23% | -0.93% | -0.45% | -0.32% | 0.13% | -0.20% |
| Class F (Screen Content) | Basketball Drill Text | -0.32% | -0.16% | 0.03% | 0.11% | -0.05% | -0.13% |
| | China Speed | 0.15% | -0.17% | 0.00% | -0.21% | -0.05% | -0.18% |
| | Slide Editing | -0.18% | -0.71% | -0.45% | 0.00% | -0.08% | -0.03% |
| | Slide Show | 0.04% | -1.35% | -0.35% | 0.18% | 0.20% | 0.04% |
| BVI-Textures (1080p) | Ball Under Water | -0.42% | 0.54% | -0.65% | -0.83% | -0.44% | -0.32% |
| | Bookcase | -0.14% | -1.03% | 0.09% | 0.01% | -0.19% | -1.22% |
| | Bricks and Bushes | -0.29% | -0.35% | -0.06% | -0.22% | -0.05% | 0.03% |
| | Bricks Leaves | -0.20% | 0.02% | -0.19% | -0.22% | -0.14% | 0.05% |
| | Bubbles Clear | 0.03% | -0.64% | -0.58% | -0.80% | -0.60% | -0.85% |
| | Calming Water | -0.42% | -1.03% | -0.68% | -1.12% | -0.63% | -0.91% |
| | Carpet | -0.58% | -0.99% | -0.60% | -1.17% | -0.39% | -1.18% |
| | Drops on Water | -0.29% | -0.55% | -0.54% | -0.90% | -0.50% | -0.47% |
| | Flowers 2 | -0.60% | -1.22% | -0.14% | -0.81% | -0.05% | -0.51% |
| | Lamp Leaves | -0.14% | -0.21% | -0.13% | -0.27% | -0.08% | -0.12% |
| | Painting Tilting | -0.30% | -0.45% | -0.31% | -0.57% | -0.29% | -0.49% |
| | Plasma Free | -0.53% | -0.76% | -1.01% | -1.42% | -0.67% | -1.19% |
| | Pond Dragonflies | -0.30% | -0.75% | -0.09% | -0.17% | -0.06% | -0.39% |
| | Smoke Clear | -0.50% | 1.04% | -0.65% | -0.68% | -0.31% | 0.49% |
| | Sparkler | -0.17% | 0.25% | -0.48% | -1.27% | -0.36% | -1.04% |
| | Squirrel | -0.28% | -0.66% | -0.12% | -0.45% | -0.25% | -0.49% |
| | Tree Willis | -0.19% | -0.28% | -0.08% | -0.04% | -0.04% | -0.05% |
| User-generated (720p) | Bike 1 | -0.37% | -0.11% | 0.07% | 0.16% | -0.35% | 0.04% |
| | Bike 2 | -0.07% | -0.18% | -0.01% | 0.56% | 0.20% | 0.68% |
| | Bike 3 | -0.30% | 1.38% | -0.17% | 0.40% | 0.34% | 0.25% |
| | Bike 4 | -0.15% | -0.76% | 0.08% | -0.07% | -0.08% | -0.13% |
| | Bike 5 | -0.44% | -0.30% | 0.11% | 0.25% | 0.02% | 0.40% |
| | Bike 6 | -0.26% | 0.01% | -0.03% | 0.19% | -0.03% | 0.47% |
| | Bike 7 | 0.24% | 0.41% | -0.20% | -0.71% | -0.01% | -0.12% |
| | Bike 8 | -0.05% | 0.07% | -0.04% | 0.28% | 0.21% | 0.36% |
| | Bike 9 | -0.06% | 0.18% | -0.06% | -0.02% | -0.20% | -0.06% |
| | Bike 10 | -0.31% | 0.55% | 0.18% | 0.70% | 0.05% | 0.14% |
| | Bike 11 | -0.15% | -0.15% | 0.30% | 0.37% | 0.19% | 0.74% |
| | Bike 12 | -0.51% | 0.07% | 0.11% | 0.26% | 0.17% | -0.01% |
| | Bike 13 | -0.14% | 0.12% | 0.03% | 0.43% | 0.05% | 0.12% |
| | Bike 14 | -0.15% | -0.36% | 0.42% | 0.76% | 0.58% | 0.41% |
| | Mittelwert | -0.21% | -0.27% | -0.14% | -0.19% | -0.07% | -0.19% |
| | Class A (4K) | -0.22% | -0.27% | -0.01% | -0.07% | -0.05% | 0.02% |
| | Class B (1080p) | -0.21% | -0.47% | -0.16% | -0.26% | -0.08% | -0.41% |
| | Class C (WVGA) | -0.05% | -0.22% | -0.07% | -0.20% | 0.01% | -0.41% |
| | Class D (WQVGA) | 0.01% | -0.27% | 0.16% | -0.07% | 0.08% | -0.37% |
| | Class E (720p) | -0.31% | -0.32% | -0.30% | -0.19% | 0.03% | 0.06% |
| | Class F (Screen Content) | -0.08% | -0.60% | -0.20% | 0.02% | 0.01% | -0.07% |
| | BVI-Textures (1080p) | -0.31% | -0.42% | -0.37% | -0.64% | -0.30% | -0.51% |
| | User-generated (720p) | -0.19% | 0.07% | 0.06% | 0.25% | 0.08% | 0.23% |

Tabelle 5.7: Verwendungshäufigkeit des vorgeschlagenen Verfahrens. Es wird angegeben, welcher Anteil der codierten Pixel mit dem CoMIC-Verfahren codiert wird. HR = Hohe Datenraten, NR = Niedrige Datenraten.

| | | All Intra | | Random Access | | Low Delay | |
|---|---|---|---|---|---|---|---|
| **Kategorie** | **Sequenz** | **HR** | **NR** | **HR** | **NR** | **HR** | **NR** |
| Class A (4K) | Traffic | 12.8% | 20.0% | 0.28% | 0.47% | 0.28% | 0.47% |
| | People on Street | 13.4% | 21.6% | 2.47% | 3.35% | 2.19% | 3.23% |
| | Nebuta | 21.8% | 24.0% | 2.18% | 0.89% | 2.00% | 0.80% |
| | Steam Locomotive | 23.9% | 26.1% | 1.60% | 1.10% | 1.30% | 1.10% |
| Class B (1080p) | Kimono | 23.2% | 30.4% | 2.96% | 3.34% | 1.64% | 2.19% |
| | Park Scene | 14.8% | 21.6% | 1.01% | 1.44% | 0.63% | 0.96% |
| | Cactus | 12.2% | 17.7% | 1.42% | 1.93% | 1.06% | 1.73% |
| | BQ Terrace | 5.53% | 9.60% | 0.24% | 0.22% | 0.17% | 0.20% |
| | Basketball Drive | 9.63% | 15.2% | 2.66% | 3.36% | 2.54% | 3.67% |
| Class C (WVGA) | Race Horses | 8.85% | 17.1% | 1.42% | 2.34% | 1.25% | 2.22% |
| | BQ Mall | 5.18% | 13.3% | 0.72% | 1.32% | 0.67% | 1.39% |
| | Party Scene | 3.78% | 11.3% | 0.96% | 1.48% | 0.86% | 1.45% |
| | Basketball Drill | 4.57% | 7.61% | 1.16% | 1.87% | 1.05% | 1.71% |
| Class D (WQVGA) | Race Horses | 10.4% | 16.8% | 2.18% | 3.18% | 1.95% | 3.16% |
| | BQ Square | 2.23% | 4.81% | 0.03% | 0.08% | 0.03% | 0.10% |
| | Blowing Bubbles | 8.45% | 20.2% | 1.44% | 1.92% | 1.52% | 2.10% |
| | Basketball Pass | 2.87% | 14.4% | 0.37% | 0.69% | 0.20% | 0.44% |
| Class E (720p) | Four People | 7.90% | 17.6% | 0.23% | 0.39% | 0.18% | 0.34% |
| | Johnny | 6.26% | 10.6% | 0.13% | 0.22% | 0.11% | 0.19% |
| | Kristen and Sara | 6.47% | 10.3% | 0.18% | 0.25% | 0.14% | 0.26% |
| Class F (Screen Content) | Basketball Drill Text | 2.88% | 5.58% | 0.93% | 1.49% | 0.85% | 1.39% |
| | China Speed | 4.27% | 7.35% | 1.85% | 1.84% | 1.74% | 1.98% |
| | Slide Editing | 0.60% | 2.57% | 0.03% | 0.10% | 0.02% | 0.08% |
| | Slide Show | 0.86% | 1.35% | 0.11% | 0.19% | 0.14% | 0.25% |
| BVI-Textures (1080p) | Ball Under Water | 4.89% | 5.27% | 6.79% | 5.91% | 7.44% | 7.65% |
| | Bookcase | 8.09% | 15.0% | 0.28% | 0.53% | 0.28% | 0.60% |
| | Bricks and Bushes | 23.7% | 32.7% | 0.93% | 1.06% | 0.55% | 0.81% |
| | Bricks Leaves | 23.1% | 32.8% | 1.04% | 1.28% | 0.94% | 1.35% |
| | Bubbles Clear | 1.07% | 2.31% | 2.14% | 2.27% | 2.01% | 2.49% |
| | Calming Water | 33.9% | 38.1% | 36.0% | 28.0% | 37.5% | 34.6% |
| | Carpet | 25.7% | 28.6% | 0.43% | 0.51% | 0.49% | 0.57% |
| | Drops on Water | 33.1% | 36.8% | 39.7% | 31.9% | 40.2% | 37.7% |
| | Flowers 2 | 26.7% | 35.6% | 0.42% | 0.60% | 0.42% | 0.59% |
| | Lamp Leaves | 19.8% | 27.8% | 2.46% | 2.80% | 1.35% | 1.93% |
| | Painting Tilting | 24.3% | 37.9% | 0.30% | 0.51% | 0.40% | 0.54% |
| | Plasma Free | 3.83% | 6.02% | 2.86% | 3.40% | 2.72% | 3.77% |
| | Pond Dragonflies | 16.3% | 25.3% | 0.23% | 0.38% | 0.25% | 0.41% |
| | Smoke Clear | 6.93% | 4.86% | 6.42% | 2.59% | 6.67% | 3.18% |
| | Sparkler | 3.61% | 7.63% | 7.59% | 9.75% | 6.95% | 9.82% |
| | Squirrel | 22.7% | 26.6% | 0.32% | 0.36% | 0.34% | 0.42% |
| | Tree Willis | 14.3% | 20.7% | 0.20% | 0.29% | 0.22% | 0.31% |
| User-generated (720p) | Bike 1 | 14.4% | 17.9% | 2.99% | 2.53% | 2.10% | 1.51% |
| | Bike 2 | 12.6% | 17.2% | 2.06% | 2.16% | 1.46% | 1.26% |
| | Bike 3 | 11.1% | 8.64% | 3.03% | 2.44% | 2.38% | 1.48% |
| | Bike 4 | 15.1% | 19.3% | 4.62% | 3.74% | 4.29% | 3.28% |
| | Bike 5 | 23.5% | 28.8% | 12.6% | 7.65% | 11.5% | 7.61% |
| | Bike 6 | 19.4% | 23.1% | 9.08% | 6.72% | 8.46% | 6.71% |
| | Bike 7 | 17.0% | 15.2% | 7.76% | 6.19% | 7.63% | 6.69% |
| | Bike 8 | 8.84% | 12.5% | 9.22% | 8.95% | 8.70% | 8.81% |
| | Bike 9 | 15.4% | 20.9% | 4.53% | 4.06% | 4.14% | 3.75% |
| | Bike 10 | 13.5% | 17.4% | 6.59% | 7.39% | 6.21% | 6.79% |
| | Bike 11 | 16.5% | 21.9% | 6.99% | 7.20% | 7.17% | 8.15% |
| | Bike 12 | 7.30% | 8.28% | 1.73% | 1.56% | 1.50% | 1.17% |
| | Bike 13 | 13.9% | 16.5% | 4.05% | 3.75% | 3.17% | 2.65% |
| | Bike 14 | 16.6% | 22.9% | 7.57% | 4.30% | 9.09% | 5.60% |
| | Mittelwert | 12.9% | 17.9% | 3.95% | 3.53% | 3.80% | 3.70% |
| | | | | | | | |
| | Class A (4K) | 18.0% | 22.9% | 1.63% | 1.46% | 1.44% | 1.40% |
| | Class B (1080p) | 13.1% | 18.9% | 1.66% | 2.06% | 1.21% | 1.75% |
| | Class C (WVGA) | 5.59% | 12.3% | 1.06% | 1.75% | 0.96% | 1.69% |
| | Class D (WQVGA) | 5.99% | 14.0% | 1.01% | 1.47% | 0.93% | 1.45% |
| | Class E (720p) | 6.88% | 12.9% | 0.37% | 0.29% | 0.15% | 0.26% |
| | Class F (Screen Content) | 2.15% | 4.21% | 0.73% | 0.91% | 0.69% | 0.93% |
| | BVI-Textures (1080p) | 17.2% | 22.6% | 6.36% | 5.42% | 6.39% | 6.28% |
| | User-generated (720p) | 14.6% | 17.9% | 5.91% | 4.90% | 5.56% | 4.68% |

Tabelle 5.8: Komplexitätssteigerung durch die vorgeschlagenen Verfahren. Die Berechnung erfolgt durch die Encoder- und Decoderlaufzeiten.

| | | All Intra | | Random Access | | Low Delay | |
|---|---|---|---|---|---|---|---|
| **Kategorie** | **Sequenz** | **Encoder** | **Decoder** | **Encoder** | **Decoder** | **Encoder** | **Decoder** |
| Class A (4K) | Traffic | 1567 | 2973 | 615 | 723 | 544 | 521 |
| | People on Street | 1239 | 4493 | 427 | 3071 | 275 | 2315 |
| | Nebuta | 3383 | 6638 | 715 | 1006 | 814 | 852 |
| | Steam Locomotive | 3663 | 7021 | 1386 | 1265 | 668 | 646 |
| Class B (1080p) | Kimono | 652 | 907 | 213 | 724 | 212 | 827 |
| | Park Scene | 1482 | 4646 | 513 | 1304 | 376 | 840 |
| | Cactus | 1246 | 2444 | 548 | 1497 | 336 | 852 |
| | BQ Terrace | 1583 | 2428 | 652 | 371 | 333 | 192 |
| | Basketball Drive | 824 | 1107 | 145 | 763 | 126 | 768 |
| Class C (WVGA) | Race Horses | 1032 | 1014 | 480 | 2583 | 256 | 1206 |
| | BQ Mall | 1392 | 798 | 492 | 785 | 587 | 1106 |
| | Party Scene | 1759 | 686 | 1362 | 1573 | 718 | 1096 |
| | Basketball Drill | 1164 | 392 | 550 | 1413 | 276 | 839 |
| Class D (WQVGA) | Race Horses | 1143 | 1005 | 427 | 1130 | 298 | 952 |
| | BQ Square | 1499 | 349 | 913 | 56 | 789 | 62 |
| | Blowing Bubbles | 1683 | 1204 | 621 | 1401 | 365 | 1052 |
| | Basketball Pass | 983 | 331 | 439 | 473 | 251 | 244 |
| Class E (720p) | Four People | 1286 | 1566 | 533 | 390 | 544 | 444 |
| | Johnny | 654 | 602 | 300 | 133 | 223 | 108 |
| | Kristen and Sara | 780 | 727 | 300 | 148 | 279 | 143 |
| Class F (Screen Content) | Basketball Drill Text | 1254 | 292 | 421 | 795 | 331 | 781 |
| | China Speed | 1438 | 344 | 549 | 702 | 442 | 697 |
| | Slide Editing | 1385 | 126 | 1099 | 55 | 929 | 66 |
| | Slide Show | 929 | 79 | 444 | 86 | 252 | 108 |
| BVI-Textures (1080p) | Ball Under Water | 336 | 173 | 74 | 668 | 71 | 903 |
| | Bookcase | 1030 | 1234 | 292 | 215 | 261 | 344 |
| | Bricks and Bushes | 2460 | 7812 | 887 | 2159 | 937 | 1446 |
| | Bricks Leaves | 2221 | 9769 | 1016 | 2427 | 1379 | 6435 |
| | Bubbles Clear | 492 | 148 | 105 | 752 | 107 | 912 |
| | Calming Water | 903 | 1730 | 164 | 5800 | 119 | 3334 |
| | Carpet | 788 | 826 | 348 | 112 | 236 | 135 |
| | Drops on Water | 956 | 2086 | 124 | 4819 | 100 | 3966 |
| | Flowers 2 | 711 | 1510 | 387 | 242 | 241 | 171 |
| | Lamp Leaves | 2155 | 5672 | 599 | 3046 | 438 | 1679 |
| | Painting Tilting | 2545 | 1834 | 1791 | 252 | 1149 | 170 |
| | Plasma Free | 886 | 1029 | 99 | 24 | 169 | 2391 |
| | Pond Dragonflies | 1563 | 3677 | 211 | 2488 | 453 | 342 |
| | Smoke Clear | 365 | 270 | 620 | 317 | 75 | 822 |
| | Sparkler | 713 | 831 | 72 | 593 | 167 | 4192 |
| | Squirrel | 1755 | 4876 | 127 | 2980 | 933 | 814 |
| | Tree Willis | 3182 | 6648 | 1257 | 555 | 1013 | 524 |
| User-generated (720p) | Bike 1 | 1012 | 1267 | 894 | 473 | 267 | 987 |
| | Bike 2 | 1268 | 2037 | 453 | 2575 | 265 | 758 |
| | Bike 3 | 507 | 301 | 283 | 1328 | 265 | 758 |
| | Bike 4 | 1045 | 1432 | 302 | 1489 | 189 | 553 |
| | Bike 5 | 954 | 1669 | 208 | 1893 | 165 | 1458 |
| | Bike 6 | 996 | 1500 | 156 | 2730 | 126 | 2415 |
| | Bike 7 | 492 | 659 | 187 | 3061 | 176 | 3075 |
| | Bike 8 | 931 | 855 | 141 | 1816 | 116 | 1714 |
| | Bike 9 | 1664 | 2813 | 148 | 2046 | 126 | 2033 |
| | Bike 10 | 821 | 863 | 322 | 3041 | 307 | 2741 |
| | Bike 11 | 664 | 1125 | 202 | 2637 | 184 | 2350 |
| | Bike 12 | 930 | 710 | 213 | 778 | 148 | 511 |
| | Bike 13 | 693 | 766 | 177 | 1210 | 124 | 765 |
| | Bike 14 | 1151 | 1999 | 163 | 1478 | 96 | 1182 |
| | Mittelwert | 1277 | 2005 | 476 | 1390 | 375 | 1211 |
| | | | | | | | |
| | Class A (4K) | 2463 | 5281 | 786 | 1516 | 575 | 1084 |
| | Class B (1080p) | 1157 | 2306 | 414 | 932 | 277 | 696 |
| | Class C (WVGA) | 1337 | 722 | 721 | 1589 | 459 | 1062 |
| | Class D (WQVGA) | 1327 | 722 | 600 | 765 | 426 | 577 |
| | Class E (720p) | 907 | 965 | 377 | 224 | 349 | 232 |
| | Class F (Screen Content) | 1252 | 210 | 628 | 410 | 489 | 413 |
| | BVI-Textures (1080p) | 1356 | 2949 | 481 | 1615 | 462 | 1681 |
| | User-generated (720p) | 938 | 1285 | 275 | 1897 | 182 | 1521 |

# 6

# ZUSAMMENFASSUNG

Die Ausgangssituation für diese Arbeit ergab sich aus der Beobachtung, dass die für die Übertragung von Videos benötigte Übertragungskapazität wesentlich schneller wächst als die hierfür zur Verfügung stehende Kanalkapazität. Hieraus entsteht die Notwendigkeit einer stetigen Verbesserung der Codierungsverfahren für die verwendeten Videocodecs.

Moderne Videocodecs beruhen in der Regel auf dem Prinzip der Hybridcodierung, also der Kombination von einer Prädiktion mit einer Transformationscodierung des Prädiktionsfehlers. Die Prädiktionsverfahren können grob in Intra- und Inter-Prädiktion unterschieden werden. Während die Intra-Codierung ausschließlich örtliche Redundanz im jeweiligen zu codierenden Bild ausnutzt, wird bei der Inter-Codierung zusätzlich die zeitliche Redundanz zwischen aufeinander folgenden Bildern mittels einer bewegungskompensierenden Prädiktion ausgenutzt. Im Rahmen dieser Arbeit sollte die Intra-Prädiktion verbessert werden.

In den eigenen Vorarbeiten [65] und [69] wurde ein als CoMIC bezeichnetes Codierungsverfahren zur Verbesserung der Intra-Prädiktion vorgeschlagen. Diese Vorarbeiten beruhen auf der Detektion von Konturen im bereits codierten Referenzbereich, der Modellierung dieser Konturen mit polynomiellen Modellen gefolgt von einer Konturextrapolation in den zu codierenden Block in Kombination mit einem als *Along-contour* bezeichneten Verfahren zur Abtastwertprädiktion, bei dem die Randabtastwerte entlang der extrapolierten Konturen fortgesetzt werden. Durch diese beiden Verfahren konnten im Gegensatz zu dem zugrunde liegenden Referenzverfahren, der Intra-Prädiktion des HEVC-Standards, mehrere unterschiedliche Konturrichtungen innerhalb eines Blocks extrapoliert werden, welche zusätzlich nicht nur linear sondern auch nichtlinear verlaufen konnten.

Im Rahmen der vorliegenden Arbeit wurde das CoMIC-Verfahren weiterentwickelt. Hierfür wurden zwei neue Verfahren vorgeschlagen:

1. Ein stochastisches Konturmodell zur Modellierung und Extrapolation der detektierten Konturen, welches das polynomielle Konturmodell ersetzt.

2. Ein auf neuronalen Netzwerken basierendes Verfahren zur Abtastwertprädiktion, welches das *Along-contour*-Verfahren ersetzt.

Der erste Beitrag in dieser Arbeit, das vorgeschlagene Verfahren zur Konturmodellierung, verarbeitet im Referenzbereich detektierte Konturen. Diese Konturen werden mit dem Canny-Algorithmus [12] detektiert. Die Hyperparameter des Canny-Algorithmus werden signaladaptiv mittels des Otsu-Verfahrens [26, 96] bestimmt. Aus dem binären Konturbild nach der Konturdetektion werden mit dem von Suzuki und Be vorgeschlagenen Verfahren [117] Vektoren mit den Koordinaten der Konturpunkte erzeugt. Für die weitere Verarbeitung werden nur Konturen berücksichtigt, die an den zu codierenden Block angrenzen.

Die Konturkoordinaten liegen nur in Ganz-Pel-Genauigkeit vor. Für eine gleichmäßigere Kontur wird versucht diese zu glätten, indem sie als kontinuierliche Funktion angenähert wird. Die Konturglättung wird nur eingesetzt, wenn der mittlere resultierende Fehler kleiner als 1 Pel ist, also im Rahmen der Konturgenauigkeit liegt. Andernfalls werden die detektierten Konturpunkte mit der gröberen Auflösung verwendet.

Die Modellierung der detektierten Kontur erfolgt mittels eines Gauß-Prozesses. Wurde die Kontur erfolgreich geglättet, dann wird ein rauschfreier Gauß-Prozess verwendet. War die Konturglättung aufgrund eines hierbei entstehenden zu großen Fehlers nicht möglich, dann wird der Gauß-Prozess unter der Annahme, dass die Konturpunkte mit Ganz-Pel-Genauigkeit aus einer Kontur mit kontinuierlichen Koordinaten durch die Überlagerung von unkorreliertem Rauschen entstanden sind, formuliert.

Der Prior-Gauß-Prozess wurde mit dem Defacto-Standard für Gauß-Prozess-Kovarianzfunktionen, dem sogenannten Squared Exponential Kernel, formuliert. In der Wahl der Kovarianzfunktion werden die Erwartungen an die zu modellierenden Konturen ausgedrückt. Die gewählte Funktion passt zu den in den durch die Partitionierung entstehenden Blöcken erwarteten glatten Konturverläufen.

Der Posterior-Gauß-Prozess ergab sich aus dem Prior-Gauß-Prozess durch Optimierung der Hyperparameter der Kovarianzfunktion für jede Kontur. Hierfür wurde die Log-Marginal-Likelihood iterativ maximiert. Für die Konturextrapolation wurde eine multivariate Gauß-Verteilung für alle Konturpunkte formuliert. Von diesen Konturpunkten sind für manche beide Koordinaten bekannt (für die detektierte Kontur) während für manche nur eine Koordinate bekannt ist (für die zu extrapolierenden Stellen). Basierend auf dieser Formulierung wurden die extrapolierten Konturpunkte als bedingte Verteilung, gegeben der Punkte der detektierten Kontur und gegeben der Stellen für die Extrapolation, formuliert. Die Prädiktion für den Konturverlauf ergibt sich aus dem Mittelwert dieser Verteilung und die Unsicherheit der Prädiktion aus der Varianz dieser Verteilung.

Der zweite Beitrag in dieser Arbeit ist ein auf neuronalen Netzwerken basierendes Verfahren zur Abtastwertprädiktion. Mit den neuronalen Netzwerken werden die benachbarten Referenzabtastwerte (vier Blöcke mit der gleichen Größe wie der zu codierende Block) sowie das Ergebnis der Konturmodellierung und -extrapolation als Eingabedaten verarbeitet, um eine Prädiktion der Abtastwerte des zu codierenden Blocks zu erzeugen. Die Konturen werden für die Abtastwertprädiktion benötigt. Die neuronalen Netzwerke wurden mit einer Autoencoder-Architektur entworfen. Autoencoder erhalten die wesentlichen Merkmale des Eingangssignals und verarbeiten diese gegebenenfalls weiter. Diese Eigenschaft passt zu der zu lösenden Aufgabe, welche darin besteht, die wesentlichen Signalcharakteristiken des Referenzbereichs zu erkennen und basierend auf ihnen das zu codierende Signal zu prädizieren.

Für das Training der neuronalen Netzwerke wurden Trainingsdaten basierend auf der RAISE-Bilddatenbank erstellt. Die Netzwerke wurden mittels der Summe der absoluten transformierten Differenzen (SATD), also der Summe der Koeffizientenbeträge nach einer Hadamard-Transformation, als Kostenfunktion trainiert. Unter der Annahme mittelwertfreier und Laplace-verteilter Prädiktionsfehlerkoeffizienten, wie sie für Videocodecs typisch sind, ist diese Kostenfunktion proportional zu der für die Übertragung des Prädiktionsfehlers benötigten Datenrate. Das prädizierte Signal, also im Fall des vorgeschlagenen Verfahrens das Ausgangssignal des neuronalen Netzwerkes, wird nie angezeigt, sondern ergibt erst zusammen mit dem übertragenen Prädiktionsfehler das letztlich angezeigte rekonstruierte Bild. Deshalb ist es bedeutender die Datenrate des Prädiktionsfehlers zu optimieren, anstatt den Fokus der Optimierung auf die Qualität des prädizierten Signals zu legen.

Für die experimentelle Untersuchung erfolgte die Integration der beiden Verfahren in den selbstentwickelten Bildcodec CoMIC sowie in die Referenzsoftware HM. Die beiden Implementierungen verfolgten unterschiedliche Ziele. Während die Implementierung in dem Bildcodec die gezielte Betrachtung einzelner Verfahren ermöglichte, konnte mit der Implementierung in der Referenzsoftware die Codierungseffizienz im Zusammenspiel mit einem modernen Videocodierungsstandard evaluiert werden. In der Referenzsoftware wurden die entwickelten Verfahren als zusätzlicher Mode implementiert, der auf CU-Ebene durch die RD-Optimierung ausgewählt werden kann, wenn er im Vergleich mit den anderen zur Verfügung stehenden Codierungsverfahren die geringsten RD-Kosten verursacht. Als Vergleichsverfahren wurden jeweils die eigenen Vorarbeiten verwendet. In diesen Vorarbeiten wurde bereits gezeigt, dass die Verfahren aus diesen Vorarbeiten besser als vergleichbare Ver-

fahren aus der Literatur sind und durch sie die Codierungseffizienz von HEVC weiter gesteigert werden konnte.

Sowohl in dem selbstentwickelten Bildcodec als auch in der Referenzsoftware des HEVC-Standards ergab sich durch die Verwendung der vorgeschlagenen Verfahren eine Steigerung der Codierungseffizienz. Unter den Laborbedingungen des Bildcodecs, in dem die vorgeschlagenen Verfahren aus dieser Arbeit jeweils mit den korrespondierenden Verfahren aus den eigenen Vorarbeiten verglichen wurden, ergab sich durch das vorgeschlagene Konturmodell eine Verbesserung um im Mittel bis zu 1,9% und durch das vorgeschlagene Verfahren zur Abtastwertprädiktion um im Mittel bis zu 8,8% - jeweils in Abhängigkeit von der Blockgröße.

Die Codierungseffizienz des Videocodecs HEVC wurde um bis zu 5% gesteigert. Gemittelt über alle 55 Testsequenzen ergaben sich für die *All Intra*-Konfiguration BD-Raten von $-0,54\%$ für hohe Datenraten und in Höhe von $-1,0\%$ für niedrige Datenraten. Durch den Einsatz des stochastischen Konturmodells anstelle des polynomiellen Konturmodells aus der eigenen Vorarbeit — jeweils in Kombination mit dem vorgeschlagenen Verfahren zur Abtastwertprädiktion — wurde eine zusätzliche Steigerung der Codierungseffizienz um $0,21$ Prozentpunkte für hohe Datenraten und um $0,27$ Prozentpunkte für niedrige Datenraten erzielt.

Die Codierungsgewinne sind für hochauflösende Videos (4K und 1080p) sowie für *User-generated Content* höher als für sehr niedrig auflösende Videos ($416\times240$) sowie computergenerierte Videos. Die höheren Codierungsgewinne für hochauflösende Videos lassen sich vermutlich auf das Vorkommen dieser Auflösungen in den Trainingsdaten der Netzwerke zurückführen. Bilder mit niedriger Auflösung sowie computergenerierte Signale kamen in den Trainingsdaten nicht vor, weswegen die geringere Codierungseffizienz hierfür nicht überraschend ist. Die höheren Gewinne für niedrige Datenraten und für *User-generated Content* lassen sich vermutlich auf die verwendete Kostenfunktion für das Training der neuronalen Netzwerke zurückführen.

Der Anstieg der Rechenzeit durch die vorgeschlagenen Verfahren ist nicht gering. Ansätze zur Beschleunigung sowohl für den Encoder als auch für den Decoder sind denkbar, beispielsweise durch eine Berechnung der Netzwerk-Inferenz auf GPUs. Zur zusätzlichen Steigerung der Codierungseffizienz gibt es noch Verbesserungspotential. Beispielsweise könnten unterschiedliche Netzwerkmodelle vorgehalten werden und adaptiv in Abhängigkeit der Charakteristik der zu codierenden Videos ausgewählt werden.

Abschließend lässt sich schlussfolgern, dass das in dieser Arbeit verfolgte Ziel, die Entwicklung einer effizienteren Intra-Codierung, durch die beiden vorgeschlagenen Verfahren erreicht wurde.

# LITERATUR


[1] Mauricio Alvarez-Mesa, Chi Ching Chi, Ben Juurlink, Valeri George und Thomas Schierl. „Parallel video decoding in the emerging HEVC standard“. In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, März 2012, S. 1545–1548. ISBN: 978-1-4673-0046-9. DOI: 10.1109/ICASSP.2012.6288186. URL: http://ieeexplore.ieee.org/document/6288186/ (siehe S. 26).

[2] Hadi Asheri, Hamid Rabiee, Nima Pourdamghani und Mohammad Ghanbari. „Multi-directional Spatial Error Concealment using Adaptive Edge Thresholding“. In: *IEEE Transactions on Consumer Electronics* 58.3 (Aug. 2012), S. 880–885. ISSN: 0098-3063. DOI: 10.1109/TCE.2012.6311331 (siehe S. 4).

[3] Oscar C. Au und S.-H. Gary Chan. „Edge-Directed Error Concealment“. In: *IEEE Transactions on Circuits and Systems for Video Technology* 20.3 (März 2010), S. 382–395. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2009.2035839 (siehe S. 4).

[4] Dzmitry Bahdanau, Kyunghyun Cho und Yoshua Bengio. „Neural Machine Translation by Jointly Learning to Align and Translate“. In: *International Conference on Machine Learning (ICLR)*. Sep. 2015. arXiv: 1409.0473 (siehe S. 33).

[5] Jim Bankoski, Paul Wilkins und Yaowu Xu. „Technical overview of VP8, an open source video codec for the web“. In: *2011 IEEE International Conference on Multimedia and Expo*. IEEE, Juli 2011, S. 1–6. ISBN: 978-1-61284-348-3. DOI: 10.1109/ICME.2011.6012227 (siehe S. 2).

[6] Eric B. Baum und David Haussler. „What Size Net Gives Valid Generalization?“ In: *Neural Computation* 1.1 (März 1989), S. 151–160. ISSN: 0899-7667. DOI: 10.1162/neco.1989.1.1.151 (siehe S. 31).

[7] Gisle Bjontegaard. *VCEG-M33: Calculation of average PSNR differences between RD-curves. ITU-T SG 16 Q 6. 13th Meeting, Austin, Texas, USA*. 2001 (siehe S. 89, 108).

[8] Gisle Bjøntegaard. *VCEG-AI11: Improvements of the BD-PSNR model. ITU-T SG 16 Q 6. 35th Meeting, Berlin, Germany*. 2008 (siehe S. 89, 108).

[9] Gisle Bjontegaard, Thomas Davies, Arild Fuldseth und Steinar Midtskogen. „The Thor Video Codec“. In: *2016 Data Compression Conference (DCC)*. IEEE, März 2016, S. 476–485. ISBN: 978-1-5090-1853-6. DOI: 10.1109/DCC.2016.74 (siehe S. 2).

[10] Frank Bossen. *JCTVC-L1100: Common test conditions and software reference configurations. 12th Meeting of the Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11. Geneva, CH*. 2013 (siehe S. 103, 108).

[11] Jill M. Boyce, Yan Ye, Jianle Chen und Adarsh K. Ramasubramonian. „Overview of SHVC: Scalable Extensions of the High Efficiency Video Coding Standard“. In: *IEEE Transactions on Circuits and Systems for Video Technology* 26.1 (Jan. 2016), S. 20–34. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2015.2461951 (siehe S. 2).

[12] John Canny. „A Computational Approach to Edge Detection“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 8.6 (Nov. 1986), S. 679–698. ISSN: 0162-8828. DOI: 10.1109/TPAMI.1986.4767851. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4767851 (siehe S. 50, 53, 120).

[13] Alfredo Canziani, Adam Paszke und Eugenio Culurciello. „An Analysis of Deep Neural Network Models for Practical Applications“. In: (Mai 2016). arXiv: 1605.07678. URL: http://arxiv.org/abs/1605.07678 (siehe S. 35).

[14] M. Augustine Cauchy. „Méthode générale pour la résolution des systèmes d'équations simultanées. Übersetzt von Richard Pulskamp, 2010.“ In: *Compte rendu des séances de l'académie des sciences* (1847), S. 536–538. URL: https://cs.uwaterloo.ca/{~}y328yu/classics/cauchy-en.pdf (siehe S. 41).

[15] Jianle Chen, Elena Alshina, Gary J. Sullivan, Jens-Rainer Ohm und Jill Boyce. *JVET E1001: Alogorithm Description of Joint Exploration Test Model 5 (JEM 5). 5th Meeting of the Joint Video Exploration Team (JVET) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11. Geneva, CH*. 2017 (siehe S. 3).

[16] Yue Chen u. a. „An Overview of Core Coding Tools in the AV1 Video Codec“. In: *2018 Picture Coding Symposium (PCS)*. IEEE, Juni 2018, S. 41–45. ISBN: 978-1-5386-4160-6. DOI: 10.1109/PCS.2018.8456249 (siehe S. 2).

[17] Yi-jen Chiu, Lidong Xu, Wenhao Zhang und Hong Jiang. „Decoder-side Motion Estimation and Wiener filter for HEVC“. In: *2013 Visual Communications and Image Processing (VCIP)*. IEEE, Nov. 2013, S. 1–6. ISBN: 978-1-4799-0290-3. DOI: 10.1109/VCIP.2013.6706446 (siehe S. 17).

[18] Cisco. *The Zettabyte Era: Trends and Analysis (white paper)*. 2017. DOI: 1465272001812119. URL: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html (siehe S. 1).

[19] Duc Tien Dang-Nguyen, Cecilia Pasquini, Valentina Conotter und Giulia Boato. „RAISE - A raw images dataset for digital image forensics“. In: *Proceedings of the 6th ACM Multimedia Systems Conference, MMSys 2015*. New York, New York, USA: Association for Computing Machinery, Inc, März 2015, S. 219–224. ISBN: 9781450333511. DOI: 10.1145/2713168.2713194. URL: http://dl.acm.org/citation.cfm?doid=2713168.2713194 (siehe S. 67, 68).

[20] Jan De Cock, Aditya Mavlankar, Anush Moorthy und Anne Aaron. „A large-scale video codec comparison of x264, x265 and libvpx for practical VOD applications“. In: *Applications of Digital Image Processing XXXIX*. Hrsg. von Andrew G. Tescher. San Francisco, CA, US: International Society for Optics und Photonics, Sep. 2016, S. 997116. DOI: 10.1117/12.2238495 (siehe S. 1, 2).

[21] Li Deng u. a. „Recent advances in deep learning for speech research at Microsoft“. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, Mai 2013, S. 8604–8608. ISBN: 978-1-4799-0356-6. DOI: 10.1109/ICASSP.2013.6639345 (siehe S. 33).

[22] Piotr Dollar und C. Lawrence Zitnick. „Fast Edge Detection Using Structured Forests“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.8 (Aug. 2015), S. 1558–1570. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2014.2377715. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6975234 (siehe S. 52).

[23] Chao Dong, Chen Change Loy, Kaiming He und Xiaoou Tang. „Image Super-Resolution Using Deep Convolutional Networks“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.2 (Feb. 2016), S. 295–307. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2015.2439281 (siehe S. 33).

[24] Sebastian Drath. *Deep Learning-basierte Intraprädiktion für die Videocodierung. Masterarbeit, Institut für Informationsverarbeitung, Leibniz Universität Hannover. Betreuer: Thorsten Laude*. 2020 (siehe S. 7, 69).

[25] David Kristjanson Duvenaud. „Automatic Model Construction with Gaussian Processes David". Diss. University of Cambridge, 2014, S. 144. URL: https://www.cs.toronto.edu/{~}duvenaud/thesis.pdf (siehe S. 58).

[26] Mei Fang, Guang Xue Yue und Qing Cang Yu. „The Study on an Application of Otsu Method in Canny Operator". In: *Proceedings of the 2009 International Symposium on Information Processing (ISIP)*. 2009 (siehe S. 55, 120).

[27] Alhussein Fawzi, Horst Samulowitz, Deepak Turaga und Pascal Frossard. „Image inpainting through neural networks hallucinations". In: *2016 IEEE 12th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)*. IEEE, Juli 2016, S. 1–5. ISBN: 978-1-5090-1929-8. DOI: 10.1109/IVMSPW.2016.7528221 (siehe S. 6, 7, 33).

[28] David Flynn, Detlev Marpe, Matteo Naccari, Tung Nguyen, Chris Rosewarne, Karl Sharman, Joel Sole und Jizheng Xu. „Overview of the Range Extensions for the HEVC Standard: Tools, Profiles, and Performance". In: *IEEE Trans. Cir. and Sys. for Video Technol.* 26.1 (Jan. 2016), S. 4–19. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2015.2478707 (siehe S. 1).

[29] Carl F. Gauß. *Theory of the combination of observations least subject to error*. 1821 (siehe S. 34).

[30] Bernd Girod. „The Efficiency of Motion-Compensating Prediction for Hybrid Coding of Video Sequences". In: *IEEE Journal on Selected Areas in Communications* 5.7 (Aug. 1987), S. 1140–1154. ISSN: 0733-8716. DOI: 10.1109/JSAC.1987.1146632 (siehe S. 2).

[31] Ian J. Goodfellow, Yoshua Bengio und Aaron Courville. *Deep Learning*. Hrsg. von Thomas Dietterich. Cambridge, MA, USA: The MIT Press, 2016 (siehe S. 5, 28–32, 38–43).

[32] Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud und Vinay Shet. „Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks". In: *International Conference on Learning Representations*. Dez. 2014. arXiv: 1312.6082 (siehe S. 33).

[33] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende und Daan Wierstra. *DRAW: A Recurrent Neural Network For Image Generation*. Feb. 2015. arXiv: 1502.04623 (siehe S. 6, 7, 33).

[34] Dan Grois, Tung Nguyen und Detlev Marpe. „Coding efficiency comparison of AV1/VP9, H.265/MPEG-HEVC, and H.264/MPEG-AVC encoders“. In: *2016 Picture Coding Symposium (PCS)*. IEEE, 2016, S. 1–5. ISBN: 978-1-5090-5966-9. DOI: 10.1109/PCS.2016.7906321 (siehe S. 1).

[35] Dan Grois, Detlev Marpe, Amit Mulayoff, Benaya Itzhaky und Ofer Hadar. „Performance comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC encoders“. In: *2013 Picture Coding Symposium (PCS)*. IEEE, Dez. 2013, S. 394–397. ISBN: 978-1-4799-0294-1. DOI: 10.1109/PCS.2013.6737766 (siehe S. 1).

[36] Kushagr Gupta, Suleman Kazi und Terry Kong. *DeepPaint: A Tool for Image Inpainting*. 2016. URL: http://cs231n.stanford.edu/reports/2016/pdfs/211{_}Report.pdf (siehe S. 6, 7, 33).

[37] Jingning Han, Ankur Saxena und Kenneth Rose. „Towards jointly optimal spatial prediction and adaptive transform in video/image coding“. In: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2010, S. 726–729. ISBN: 978-1-4244-4295-9. DOI: 10.1109/ICASSP.2010.5495043. URL: http://ieeexplore.ieee.org/document/5495043/ (siehe S. 24).

[38] Philippe Hanhart, Martin Rerabek, Francesca De Simone und Touradj Ebrahimi. „Subjective quality evaluation of the upcoming HEVC video compression standard“. In: *SPIE Optical Engineering + Applications*. Okt. 2012, S. 84990V. DOI: 10.1117/12.946036 (siehe S. 1).

[39] Donald O. Hebb. *The Organization of Behavior*. New York, NY, USA: Wiley, 1949 (siehe S. 34).

[40] Geoffrey Hinton u. a. „Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups“. In: *IEEE Signal Processing Magazine* 29.6 (Nov. 2012), S. 82–97. ISSN: 1053-5888. DOI: 10.1109/MSP.2012.2205597 (siehe S. 33).

[41] Daniel Holden, Jun Saito und Taku Komura. „A deep learning framework for character motion synthesis and editing“. In: *ACM Transactions on Graphics* 35.4 (Juli 2016), S. 1–11. ISSN: 07300301. DOI: 10.1145/2897824.2925975 (siehe S. 33).

[42] Kurt Hornik, Maxwell Stinchcombe und Halbert White. „Multilayer feedforward networks are universal approximators“. In: *Neural Networks* 2.5 (Jan. 1989), S. 359–366. ISSN: 0893-6080. DOI: 10.1016/0893-6080(89)90020-8. URL: https://www.sciencedirect.com/science/article/pii/0893608089900208 (siehe S. 41).

[43] Farzad Husain, Babette Dellen und Carme Torras. „Scene Understanding Using Deep Learning“. In: *Handbook of Neural Computation*. Hrsg. von Pijush Samui, Sanjiban Sekhar und Valentina E. Balas. Academic Press, Jan. 2017, S. 373–382. ISBN: 9780128113189. DOI: 10.1016/B978-0-12-811318-9.00020-X. URL: https://www.sciencedirect.com/science/article/pii/B978012811318900020X (siehe S. 33).

[44] ISO/IEC 14496–10. *Coding of Audiovisual Objects-Part 10: Advanced Video Coding/ITU-T Recommendation H.264 Advanced video coding for generic audiovisual services*. 2003 (siehe S. 1).

[45] ISO/IEC JTC 1/SC 29. *ISO/IEC 11544: Information technology - Coded representation of picture and audio information - Progressive bi-level image compression*. Techn. Ber. 1993, S. 1–77 (siehe S. 4).

[46] ISO/IEC. *10918-1 - Information technology - Digital compression and coding of continuous-tone still images*. 1994 (siehe S. 4).

[47] *ITU-T Recommendation H.265/ ISO/IEC 23008-2:2013 MPEG-H Part 2: High Efficiency Video Coding (HEVC)*. 2013 (siehe S. 1, 19).

[48] ITU-T. *Recommendation T.81 - Information technology - Digital compression and coding of continuous-tone still images*. 1992 (siehe S. 4).

[49] ITU-T. *Recommendation T.82 : Information technology - Coded representation of picture and audio information - Progressive bi-level image compression*. 1993 (siehe S. 4).

[50] Sergey Ioffe und Christian Szegedy. „Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift“. In: *Proceedings of The 32nd International Conference on Machine Learning*. 2015, S. 448–456 (siehe S. 6, 77).

[51] Alexey G. Ivakhnenko. „Polynomial theory of complex systems“. In: *Transactions on Systems, Man and Cybernetics* 4 (1971), S. 364–378 (siehe S. 34).

[52] Alexey G. Ivakhnenko und Valentin G. Lapa. „Cybernetic Predicting Devices“. In: *CCM Information Corporation* (1965) (siehe S. 34).

[53] Justin Johnson, Alexandre Alahi und Li Fei-Fei. „Perceptual Losses for Real-Time Style Transfer and Super-Resolution“. In: *European Conference on Computer Vision (ECCV)*. Springer, Cham, 2016, S. 694–711. DOI: 10.1007/978-3-319-46475-6_43 (siehe S. 33).

[54] Joint Video Exploration Team (JVET) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11. *JVET-G1010: JVET common test conditions and software reference configurations. 7th Meeting: Torino, IT, 13–21 July*. 2017 (siehe S. 27).

[55] Jiwon Kim, Jung Kwon Lee, Kyoung Mu Lee, Jung Kwon Lee und Kyoung Mu Lee. „Accurate Image Super-Resolution Using Very Deep Convolutional Networks". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Juni 2016, S. 1646–1654. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.182 (siehe S. 33).

[56] Diederik Kingma und Jimmy Ba. „Adam: A Method for Stochastic Optimization". In: *ArXiv preprint: 1412.6980* (Dez. 2014). arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980 (siehe S. 78).

[57] Sven Klomp. *Decoderseitige Bewegungsschätzung in der Videocodierung*. Düsseldorf: Fortschritt-Berichte VDI : Reihe 10, Informatik, Kommunikation, VDI Verlag, 2012, S. 1–98. ISBN: 978-3-18-382010-8 (siehe S. 17).

[58] Sven Klomp, Marco Munderloh und Jorn Ostermann. „Block size dependent error model for motion compensation". In: *2010 IEEE International Conference on Image Processing*. IEEE, Sep. 2010, S. 969–972. ISBN: 978-1-4244-7992-4. DOI: 10.1109/ICIP.2010.5649414 (siehe S. 17).

[59] Sven Klomp, Marco Munderloh und Jörn Ostermann. „Decoder-Side Motion Estimation Assuming Temporally or Spatially Constant Motion". In: *ISRN Signal Processing* 2011 (Juni 2011), S. 1–10. ISSN: 2090-5041. DOI: 10.5402/2011/956372 (siehe S. 17).

[60] Jonas Kohler, Hadi Daneshmand, Aurelien Lucchi, Ming Zhou, Klaus Neymeyr und Thomas Hofmann. „Exponential convergence rates for Batch Normalization: The power of length-direction decoupling in non-convex optimization". In: (Mai 2018). arXiv: 1805.10694. URL: http://arxiv.org/abs/1805.10694 (siehe S. 77).

[61] Alex Krizhevsky, Ilya Sutskever und Geoffrey E. Hinton. „ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. 2012, S. 1097–1105 (siehe S. 6, 32, 34).

[62] Jani Lainema und Woo-Jin Han. „Intra-Picture Prediction in HEVC". In: *High Efficiency Video Coding (HEVC): Algorithms and Architectures*. Hrsg. von Vivienne Sze, Madhukar Budagavi und Gary J. Sullivan. Cham: Springer International Publishing, 2014, S. 91–112. ISBN: 978-3-319-06895-4. DOI: 10.1007/978-3-319-06895-4_4.

URL: https://doi.org/10.1007/978-3-319-06895-4{_}4 (siehe S. 24).

[63] Jani Lainema, Frank Bossen, Woo-Jin Han, Junghye Min und Kemal Ugur. „Intra Coding of the HEVC Standard". In: *IEEE Transactions on Circuits and Systems for Video Technology* 22.12 (Dez. 2012), S. 1792–1801. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2012.2221525 (siehe S. 3, 16, 22, 23).

[64] Stéphane Lathuilière, Pablo Mesejo, Xavier Alameda-Pineda und Radu Horaud. „A Comprehensive Analysis of Deep Regression". In: (März 2018). arXiv: 1803.08450. URL: http://arxiv.org/abs/1803.08450 (siehe S. 32).

[65] Thorsten Laude und Jörn Ostermann. „Contour-based Multidirectional Intra Coding for HEVC". In: *Proceedings of 32nd Picture Coding Symposium (PCS)*. Nuremberg, Germany: IEEE, 2016. DOI: 10.1109/PCS.2016.7906319 (siehe S. 5–7, 9, 53, 55, 65, 93, 119).

[66] Thorsten Laude und Jörn Ostermann. „Deep learning-based intra prediction mode decision for HEVC". In: *Proceedings of 32nd Picture Coding Symposium (PCS)*. Nuremberg, Germany: IEEE, 2016. DOI: 10.1109/PCS.2016.7906399 (siehe S. 27, 32, 71).

[67] Thorsten Laude, Yannick Richter und Jörn Ostermann. „Neural Network Compression using Transform Coding and Clustering". In: *NIPS Compact Deep Neural Network Representation with Industrial Applications Workshop*. Montreal, Canada, Dez. 2018. arXiv: 1805.07258 (siehe S. 49).

[68] Thorsten Laude, Holger Meuel, Yiqun Liu und Jorn Ostermann. „Motion blur compensation in scalable HEVC hybrid video coding". In: *2013 Picture Coding Symposium (PCS)*. San Jose, CA, USA: IEEE, 2013, S. 313–316. ISBN: 978-1-4799-0294-1. DOI: 10.1109/PCS.2013.6737746. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6737746 (siehe S. 105).

[69] Thorsten Laude, Jan Tumbrägel, Marco Munderloh und Jörn Ostermann. „Non-linear contour-based multidirectional intra coding". In: *APSIPA Transactions on Signal and Information Processing* 7.11 (Okt. 2018). ISSN: 2048-7703. DOI: 10.1017/ATSIP.2018.14 (siehe S. 2, 3, 5–7, 9, 53, 55, 59, 64, 65, 89, 93, 109, 119).

[70] Thorsten Laude, Yeremia Gunawan Adhisantoso, Jan Voges, Marco Munderloh und Jörn Ostermann. „A Comprehensive Video Codec Comparison". In: *APSIPA Transactions on Signal and Information Processing* 8 (Nov. 2019), e30. ISSN: 2048-7703. DOI: 10.1017/ATSIP.2019.23 (siehe S. 105).

[71] Yann LeCun, Yoshua Bengio und Geoffrey Hinton. „Deep learning“. In: *Nature* 521.7553 (Mai 2015), S. 436–444. ISSN: 0028-0836. DOI: 10.1038/nature14539 (siehe S. 5).

[72] Christian Ledig u. a. „Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network“. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Juli 2017, S. 105–114. ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.19 (siehe S. 33).

[73] Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus und Shimon Schocken. „Multilayer feedforward networks with a nonpolynomial activation function can approximate any function“. In: *Neural Networks* 6.6 (Jan. 1993), S. 861–867. ISSN: 0893-6080. DOI: 10.1016/S0893-6080(05)80131-5. URL: https://www.sciencedirect.com/science/article/pii/S0893608005801315 (siehe S. 41).

[74] Jian-Liang Lin, Yi-Wen Chen, Yu-Wen Huang und Shaw-Min Lei. „Motion Vector Coding in the HEVC Standard“. In: *IEEE Journal of Selected Topics in Signal Processing* 7.6 (Dez. 2013), S. 957–968. ISSN: 1932-4553. DOI: 10.1109/JSTSP.2013.2271975 (siehe S. 17).

[75] Dong Liu, Xiaoyan Sun und Feng Wu. „Edge-Based Inpainting and Texture Synthesis for Image Compression“. In: *IEEE International Conference on Multimedia and Expo (ICME)*. Juli 2007, S. 1443–1446. ISBN: 1-4244-1016-9. DOI: 10.1109/ICME.2007.4284932 (siehe S. 4–6, 8).

[76] Dong Liu, Xiaoyan Sun, Feng Wu, Shipeng Li und Ya-Qin Zhang. „Image Compression With Edge-Based Inpainting“. In: *IEEE Transactions on Circuits and Systems for Video Technology* 17.10 (Okt. 2007), S. 1273–1287. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2007.903663 (siehe S. 4, 6, 8).

[77] Dong Liu, Xiaoyan Sun, Feng Wu und Ya-Qin Zhang. „Edge-oriented Uniform Intra Prediction.“ In: *IEEE Transactions on Image Processing* 17.10 (Okt. 2008), S. 1827–36. ISSN: 1057-7149. DOI: 10.1109/TIP.2008.2002835 (siehe S. 4–6, 8).

[78] Christian Lottermann und Eckehard Steinbach. „Modeling the bit rate of H.264/AVC video encoding as a function of quantization parameter, frame rate and GoP characteristics“. In: *2014 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*. IEEE, Juli 2014, S. 1–6. ISBN: 978-1-4799-4717-1. DOI: 10.1109/ICMEW.2014.6890567 (siehe S. 4).

[79] Bradley J. Lucier. *True Color Kodak Images*. URL: http://r0k.us/graphics/kodak/ (besucht am 17. 04. 2020) (siehe S. 87).

[80] Detlef Marpe, Thomas Wiegand und Gary J. Sullivan. „The H.264/MPEG4 advanced video coding standard and its applications“. In: *IEEE Communications Magazine* 44.8 (Aug. 2006), S. 134–143. ISSN: 0163-6804. DOI: 10.1109/MCOM.2006.1678121 (siehe S. 1).

[81] W. McCulloch und W. Pitts. „A logical calculus of the ideas immanent in nervous activity“. In: *Bulletin of Mathematical Biophysics* 7 (1943), S. 115–133 (siehe S. 34).

[82] J. Min, S. Lee, I. Kim, W.-J. Han, J. Lainema und K. Ugur. *Unification of the Directional Intra Prediction Methods in TMuC, JCTVC-B100, Geneva, Switzerland, July*. 2010 (siehe S. 3, 23).

[83] Richard von Mises. *Mathematical Theory of Probability and Statistics*. Hrsg. von Hilda Geiringer. New York und London: Academic Press, 1964 (siehe S. 62).

[84] Kiran Misra, Andrew Segall, Michael Horowitz, Shilin Xu, Arild Fuldseth und Minhua Zhou. „An Overview of Tiles in HEVC“. In: *IEEE Journal of Selected Topics in Signal Processing* 7.6 (Dez. 2013), S. 969–977. ISSN: 1932-4553. DOI: 10.1109/JSTSP.2013.2271451 (siehe S. 21).

[85] Tom M. Mitchell. *Machine Learning*. New York, NY, USA: McGraw-Hill, 1997. ISBN: 0070428077 (siehe S. 28).

[86] Debargha Mukherjee, Jim Bankoski, Adrian Grange, Jingning Han, John Koleszar, Paul Wilkins, Yaowu Xu und Ronald Bultje. „The latest open-source video codec VP9 - An overview and preliminary results“. In: *2013 Picture Coding Symposium (PCS)*. IEEE, Dez. 2013, S. 390–393. ISBN: 978-1-4799-0294-1. DOI: 10.1109/PCS.2013.6737765 (siehe S. 2).

[87] Debargha Mukherjee, Hui Su, James Bankoski, Alex Converse, Jingning Han, Zoe Liu und Yaowu Xu. „An overview of new video coding tools under consideration for VP10: the successor to VP9“. In: *SPIE Optical Engineering + Application* 9599 (Sep. 2015). Hrsg. von Andrew G. Tescher. DOI: 10.1117/12.2191104 (siehe S. 2).

[88] Matthias Narroschke. *Adaptive Prädiktionsfehlercodierung für die Hybridcodierung von Videosignalen*. Fortschritt-Berichte VDI Reihe 10 Nr. 786. Düsseldorf: VDI Verlag, 2008. ISBN: 978-3-18-378610-7 (siehe S. 24, 80, 105).

[89] Netflix. *Internet Connection Speed Recommendations (https://help.netflix.com/en/node/306, abgerufen am 06.05.2020)*. 2020 (siehe S. 105).

[90] Andrew Y. Ng und Michael I. Jordan. „On Discriminative vs. Generative Classifiers: A comparison of logistic regression and naive Bayes“. In: *Advances in Neural Information Processing Systems 14 (NIPS 2001)*. Hrsg. von T.G. Dietterich, S. Becker und Z. Ghahramani. Neural Information Processing Systems Foundation, Inc., 2001, S. 841–848 (siehe S. 29).

[91] Tung Nguyen, Philipp Helle, Martin Winken, Benjamin Bross, Detlev Marpe, Heiko Schwarz und Thomas Wiegand. „Transform Coding Techniques in HEVC“. In: *IEEE Journal of Selected Topics in Signal Processing* 7.6 (Dez. 2013), S. 978–989. ISSN: 1932-4553. DOI: 10.1109/JSTSP.2013.2278071 (siehe S. 24).

[92] Jens-Rainer Ohm. *Multimedia Communication Technology : Representation, Transmission and Identification of Multimedia Signals*. Springer Berlin Heidelberg, 2004, S. 859. ISBN: 9783642187506. DOI: 10.1007/978-3-642-18750-6 (siehe S. 2).

[93] Jens-Rainer Ohm und Mathias Wien. „Status and Perspectives of Video Coding“. In: *International Brodcasting Convention* (2017) (siehe S. 3).

[94] Jens-Rainer Ohm, Gary J. Sullivan, Heiko Schwarz, Thiow Keng Tan und Thomas Wiegand. „Comparison of the Coding Efficiency of Video Coding Standards—Including High Efficiency Video Coding (HEVC)“. In: *IEEE Transactions on Circuits and Systems for Video Technology* 22.12 (Dez. 2012), S. 1669–1684. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2012.2221192 (siehe S. 1).

[95] Jörn Ostermann, Jan Bormans, Peter List, Detlef Marpe, Matthias Narroschke, Fernando Pereira, Thomas Stockhammer und Thomas Wedi. „Video coding with H.264/AVC: tools, performance, and complexity“. In: *IEEE Circuits and Systems Magazine* 4.1 (2004), S. 7–28. ISSN: 1531-636X. DOI: 10.1109/MCAS.2004.1286980 (siehe S. 1, 16).

[96] Nobuyuki Otsu. „A Threshold Selection Method from Gray-Level Histograms“. In: *IEEE Transactions on Systems, Man and Cybernetics* 9.1 (1979), S. 62–66 (siehe S. 55, 120).

[97] Miltiadis Alexios Papadopoulos, Fan Zhang, Dimitris Agrafiotis und David Bull. „A Video Texture Database for Perceptual Compression and Quality Assessment“. In: *International Conference on Image Processing (ICIP)*. Quebec, Canada, 2015 (siehe S. 105).

[98] Vilfredo Pareto und Alfred N. Page. *Manuale di economia politica ("Handbuch der politischen Ökonomie")*. Hrsg. von A.M. Kelley. 1971. ISBN: 978-0-678-00881-2 (siehe S. 32).

[99] Omkar M Parkhi, Andrea Vedaldi und Andrew Zisserman. „Deep Face Recognition“. In: *British Machine Vision Conference*. Swansea, UK, 2015 (siehe S. 32).

[100] Carl Edward Rasmussen und Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press MIT Press, 2006. URL: http://www.gaussianprocess.org/gpml/chapters/RW.pdf (siehe S. 57, 58, 60–62).

[101] Christoph Reinders, Hanno Ackermann, Michael Ying Yang und Bodo Rosenhahn. „Object Recognition from very few Training Examples for Enhancing Bicycle Maps“. In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, Juni 2018, S. 1–8. ISBN: 978-1-5386-4452-2. DOI: 10.1109/IVS.2018.8500469. URL: https://ieeexplore.ieee.org/document/8500469/ (siehe S. 48).

[102] Frank Rosenblatt. „The perceptron: a probabilistic model for information storage and organization in the brain“. In: *Psychological review* 65.6 (1958), S. 386 (siehe S. 34).

[103] Olga Russakovsky u. a. „ImageNet Large Scale Visual Recognition Challenge“. In: (Sep. 2014). arXiv: 1409.0575 (siehe S. 32).

[104] Arthur L. Samuel. „Some studies in machine learning using the game of checkers“. In: *IBM Journal of Research and Development* 3.3 (Jan. 1959). ISSN: 0018-8646. DOI: 10.1147/rd.441.0206 (siehe S. 27).

[105] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas und Aleksander Madry. „How Does Batch Normalization Help Optimization?“ In: *Advances in Neural Information Processing Systems 31*. Hrsg. von S. Bengio and H. Wallach and H. Larochelle and K. Grauman and N. Cesa-Bianchi and R. Garnett. Montreal, Canada, 2018, S. 2487–2497. URL: http://papers.nips.cc/paper/7515-how-does-batch-normalization-help-optimization (siehe S. 77).

[106] Jürgen Schmidhuber. „Deep learning in neural networks: An overview“. In: *Neural Networks* (Okt. 2014), S. 88. ISSN: 08936080. DOI: 10.1016/j.neunet.2014.09.003. arXiv: 1404.7828 (siehe S. 31, 34).

[107] Heiko Schwarz, Detlev Marpe und Thomas Wiegand. „Analysis of Hierarchical B Pictures and MCTF“. In: *2006 IEEE International Conference on Multimedia and Expo*. IEEE, Juli 2006, S. 1929–1932. ISBN: 1-4244-0366-7. DOI: 10.1109/ICME.2006.262934. URL: http://ieeexplore.ieee.org/document/4037003/ (siehe S. 16).

[108] Heiko Schwarz, Thomas Schierl und Detlev Marpe. „Block Structures and Parallelism Features in HEVC“. In: 2014, S. 49–90. DOI: 10.1007/978-3-319-06895-4_3 (siehe S. 21).

[109] Kasiviswanathan S. Shanmugam. „Comments on Discrete Cosine Transform“. In: *IEEE Transactions on Computers* C-24.7 (Juli 1975), S. 759–759. ISSN: 0018-9340. DOI: 10.1109/T-C.1975.224301 (siehe S. 18).

[110] Or Sharir, Barak Peleg und Yoav Shoham. „The Cost of Training NLP Models: A Concise Overview“. In: (2020). arXiv: 2004.08900. URL: http://arxiv.org/abs/2004.08900 (siehe S. 48).

[111] Rickard Sjoberg, Ying Chen, Akira Fujibayashi, Miska M. Hannuksela, Jonatan Samuelsson, Thiow Keng Tan, Ye-Kui Wang und Stephan Wenger. „Overview of HEVC High-Level Syntax and Reference Picture Management“. In: *IEEE Transactions on Circuits and Systems for Video Technology* 22.12 (Dez. 2012), S. 1858–1870. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2012.2223052 (siehe S. 22).

[112] Irwin Sobel. *An Isotropic 3x3 Image Gradient Operator*. Feb. 2014 (siehe S. 50).

[113] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever und Ruslan Salakhutdinov. „Dropout: a simple way to prevent neural networks from overfitting“. In: *The Journal of Machine Learning Research* 15.1 (2014), S. 1929–1958. ISSN: 1532-4435 (siehe S. 6, 80, 82).

[114] Jacob Ström, Kenneth Andersson, Rickard Sjöberg, Andrew Segall, Frank Bossen, Gary J. Sullivan und Jens-Rainer Ohm. *JVET-Q2016-v4: Summary information on BD-rate experiment evaluation practices. 17th Meeting of the Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11. Brussels, BE*. 2020 (siehe S. 108).

[115] Gary J. Sullivan und Thomas Wiegand. „Rate-distortion optimization for video compression“. In: *IEEE Signal Processing Magazine* 15.6 (1998), S. 74–90. ISSN: 10535888. DOI: 10.1109/79.733497 (siehe S. 27).

[116] Gary J Sullivan, Jens-Rainer Ohm, Woo-jin Han und Thomas Wiegand. „Overview of the High Efficiency Video Coding (HEVC) Standard“. In: *IEEE Transactions on Circuits and Systems for Video Technology* 12 (2012) (siehe S. 1, 19).

[117] Satoshi Suzuki und Keiichia Be. „Topological Structural Analysis of Digitized Binary Images by Border Following“. In: *Computer Vision, Graphics, and Image Processing* 30.1 (Apr. 1985), S. 32–46. ISSN: 0734189X. DOI: 10.1016/0734-189X(85)90016-7. URL: http://www.sciencedirect.com/science/article/pii/0734189X85900167 (siehe S. 55, 120).

[118] Vivienne Sze und Madhukar Budagavi. „High Throughput CABAC Entropy Coding in HEVC“. In: *IEEE Transactions on Circuits and Systems for Video Technology* 22.12 (Dez. 2012), S. 1778–1791. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2012.2221526 (siehe S. 19).

[119] Vivienne Sze, Madhukar Budagavi und Gary J. Sullivan, Hrsg. *High Efficiency Video Coding (HEVC)*. Integrated Circuits and Systems. Cham: Springer International Publishing, 2014. ISBN: 978-3-319-06894-7. DOI: 10.1007/978-3-319-06895-4 (siehe S. 1).

[120] Vivienne Sze und Detlev Marpe. „Entropy Coding in HEVC“. In: *High Efficiency Video Coding (HEVC): Algorithms and Architectures*. Hrsg. von Vivienne Sze, Madhukar Budagavi und Gary J. Sullivan. Heidelberg, 2014, S. 209–274. ISBN: 978-3-319-06894-7. DOI: 10.1007/978-3-319-06895-4_8 (siehe S. 26).

[121] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, Joel S. Emer, Jian-Hao Luo, Jianxin Wu und Weiyao Lin. „Efficient Processing of Deep Neural Networks: A Tutorial and Survey“. In: *Proceedings of the IEEE* 105.12 (Dez. 2017), S. 2295–2329. ISSN: 0018-9219. DOI: 10.1109/JPROC.2017.2761740 (siehe S. 35, 36).

[122] Christian Szegedy, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke und Andrew Rabinovich. „Going deeper with convolutions“. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Juni 2015, S. 1–9. ISBN: 978-1-4673-6964-0. DOI: 10.1109/CVPR.2015.7298594 (siehe S. 6).

[123] The HDF Group. *Hierarchical data format version 5*. 2000-2010. URL: http://www.hdfgroup.org/HDF5 (siehe S. 70).

[124] Lucas Theis und Matthias Bethge. „Generative Image Modeling Using Spatial LSTMs“. In: *NIPS*. MIT Press Cambridge, 2015, S. 1927–1935 (siehe S. 6, 7, 33).

[125] George Toderici, Sean M. O'Malley, Sung Jin Hwang, Damien Vincent, David Minnen, Shumeet Baluja, Michele Covell und Rahul Sukthankar. *Variable Rate Image Compression with Recurrent Neural Networks*. Nov. 2015. arXiv: 1511.06085 (siehe S. 2, 7).

[126] George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor und Michele Covell. *Full Resolution Image Compression with Recurrent Neural Networks*. Aug. 2016. arXiv: 1608.05148 (siehe S. 2, 7).

[127] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun und Christoph Bregler. „Efficient object localization using Convolutional Networks". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Bd. 07-12-June-2015. IEEE Computer Society, Okt. 2015, S. 648–656. ISBN: 9781467369640. DOI: 10.1109/CVPR.2015.7298664. arXiv: 1411.4280 (siehe S. 82).

[128] Alexander Toshev und Christian Szegedy. „DeepPose: Human Pose Estimation via Deep Neural Networks". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2014, S. 1653–1660. DOI: 10.1109/CVPR.2014.214 (siehe S. 32).

[129] Jan Tumbrägel. *Nichtlineare Kantenextrapolation für die Prädiktion in der Bildcodierung. Diplomarbeit, Institut für Informationsverarbeitung, Leibniz Universität Hannover, Betreuer: Jörn Ostermann*. 2016 (siehe S. 6).

[130] Jean-Marc Valin, Timothy B. Terriberry, Nathan E. Egge, Thomas Daede, Yushin Cho, Christopher Montgomery und Michael Bebenita. *Daala: Building A Next-Generation Video Codec From Unconventional Technology*. Aug. 2016. arXiv: 1608.01947 (siehe S. 2).

[131] Fjodor van Veen. *The Neural Network Zoo*. 2016. URL: https://www.asimovinstitute.org/neural-network-zoo/ (besucht am 18. 02. 2019) (siehe S. 39).

[132] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio und Pierre-Antoine Manzagol. „Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion". In: *Journal of Machine Learning Research* 11.12 (2010), S. 3371–3408. ISSN: ISSN 1533-7928 (siehe S. 33).

[133] Gregory K. Wallace. „The JPEG Still Picture Compression Standard". In: *IEEE Transactions on Consumer Electronics* 38.1 (1992). DOI: 10.1109/30.125072 (siehe S. 4).

[134] Xiaolong Wang, Rui Guo und Chandra Kambhamettu. „Deeply-Learned Feature for Age Estimation". In: *2015 IEEE Winter Conference on Applications of Computer Vision*. IEEE, Jan. 2015, S. 534–541. ISBN: 978-1-4799-6683-7. DOI: 10.1109/WACV.2015.77 (siehe S. 32).

[135] Allan G. Weber. *SIPI Image Database*. URL: http://sipi.usc.edu/database/ (besucht am 17. 04. 2020) (siehe S. 87).

[136] Allan G. Weber. *The USC-SIPI Image Database: Version 6*. Techn. Ber. 2018. URL: http://netpbm.sourceforge.net/ (siehe S. 87).

[137] Paul J. Werbos. „Applications of advances in nonlinear sensitivity analysis“. In: *Proceedings of the 10th IFIP Conference*. 1981, S. 762–770 (siehe S. 34).

[138] Bernard Widrow und Marcian Hoff. „Associative storage and retrieval of digital information in networks of adaptive neurons“. In: *Biological Prototypes and Synthetic Systems* 1 (1962), S. 160 (siehe S. 34).

[139] Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard und Ajay Luthra. „Overview of the H.264/AVC video coding standard“. In: *IEEE Transactions on Circuits and Systems for Video Technology* 13.7 (Juli 2003), S. 560–576. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2003.815165. URL: http://ieeexplore.ieee.org/document/1218189/ (siehe S. 16).

[140] Mathias Wien. „High Efficiency Video Coding - Coding Tools and Specification“. In: *Springer* (2015) (siehe S. 1, 12, 19–21, 23, 104).

[141] David H. Wolpert und William G. Macready. „No Free Lunch Theorems for Optimization“. In: *Trans. Evol. Comp* 1.1 (Apr. 1997), S. 67–82. ISSN: 1089-778X. DOI: 10.1109/4235.585893 (siehe S. 32).

[142] Lonce Wyse. „Audio Spectrogram Representations for Processing with Convolutional Neural Networks“. In: *Proceedings of the First International Conference on Deep Learning and Music*. Anchorage, US, Juni 2017, S. 37–41. arXiv: 1706.09559. URL: http://arxiv.org/abs/1706.09559 (siehe S. 71).

[143] Junyuan Xie, Linli Xu und Enhong Chen. „Image Denoising and Inpainting with Deep Neural Networks“. In: *Conference on Neural Information Processing Systems (NIPS)*. 2012, S. 341–349 (siehe S. 33).

[144] Jizheng Xu, Rajan Joshi und Robert A. Cohen. „Overview of the Emerging HEVC Screen Content Coding Extension“. In: *IEEE Transactions on Circuits and Systems for Video Technology* 26.1 (Jan. 2016), S. 50–62. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2015.2478706 (siehe S. 2, 24).

[145] Xiaozhong Xu, Shan Liu, Tzu-Der Chuang, Yu-Wen Huang, Shaw-Min Lei, Krishnakanth Rapaka, Chao Pang, Vadim Seregin, Ye-Kui Wang und Marta Karczewicz. „Intra Block Copy in HEVC Screen Content Coding Extensions“. In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 6.4 (Dez. 2016), S. 409–419. ISSN: 2156-3357. DOI: 10.1109/JETCAS.2016.2597645 (siehe S. 16).

[146] Yule Yuan und Xiaohang Sun. „Edge Information Based Effective Intra Mode Decision Algorithm“. In: *2012 IEEE International Conference on Signal Processing, Communication and Computing (ICSPCC)*. IEEE, Aug. 2012, S. 628–633. ISBN: 978-1-4673-2193-8. DOI: 10.1109/ICSPCC.2012.6335602 (siehe S. 4).

[147] Soumaya Zaghbani, Noureddine Boujneh und Med Salim Bouhlel. „Age estimation using deep learning“. In: *Computers & Electrical Engineering* 68 (Mai 2018), S. 337–347. ISSN: 0045-7906. DOI: 10.1016/J.COMPELECENG.2018.04.012 (siehe S. 32).

[148] Heiga Ze, Andrew Senior und Mike Schuster. „Statistical parametric speech synthesis using deep neural networks“. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, Mai 2013, S. 7962–7966. ISBN: 978-1-4799-0356-6. DOI: 10.1109/ICASSP.2013.6639215 (siehe S. 33).

[149] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng und Lei Zhang. „Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising“. In: *IEEE Transactions on Image Processing* 26.7 (Juli 2017), S. 3142–3155. ISSN: 1057-7149. DOI: 10.1109/TIP.2017.2662206 (siehe S. 33).

[150] Li Zhang, Xiaoyu Xiu, Jianle Chen, Marta Karczewicz, Yunwen He, Yan Ye, Jizheng Xu, Joel Sole und Woo-Shik Kim. „Adaptive Color-Space Transform in HEVC Screen Content Coding“. In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 6.4 (Dez. 2016), S. 446–459. ISSN: 2156-3357. DOI: 10.1109/JETCAS.2016.2599860. URL: http://ieeexplore.ieee.org/document/7553470/ (siehe S. 11).

[151] Barret Zoph und Quoc V. Le. „Neural Architecture Search with Reinforcement Learning“. In: *ICLR*. Nov. 2017. arXiv: 1611.01578 (siehe S. 48).

# VERÖFFENTLICHUNGEN

Während der Zeit am Institut für Informationsverarbeitung sind die folgenden Veröffentlichungen entstanden:

## ZEITSCHRIFTENARTIKEL


[1] Thorsten Laude, Jan Tumbrägel, Marco Munderloh und Jörn Ostermann. „Non-linear contour-based multidirectional intra coding". In: *APSIPA Transactions on Signal and Information Processing* 7.11 (Okt. 2018). ISSN: 2048-7703. DOI: 10.1017/ATSIP.2018.14.

[2] Thorsten Laude, Yeremia Gunawan Adhisantoso, Jan Voges, Marco Munderloh und Jörn Ostermann. „A Comprehensive Video Codec Comparison". In: *APSIPA Transactions on Signal and Information Processing* 8 (Nov. 2019), e30. ISSN: 2048-7703. DOI: 10.1017/ATSIP.2019.23.


## KONFERENZBEITRÄGE


[1] Thorsten Laude, Felix Haub und Jörn Ostermann. „HEVC Inter Coding using Deep Recurrent Neural Networks and Artificial Reference Pictures". In: *Picture Coding Symposium (PCS)*. Nov. 2019.

[2] Thorsten Laude und Jörn Ostermann. „Copy Mode for Static Screen Content Coding with HEVC". In: *IEEE International Conference on Image Processing (ICIP)*. Sep. 2015, S. 1930 –1934. DOI: 10.1109/ICIP.2015.7351137.

[3] Thorsten Laude und Jörn Ostermann. „Contour-based Multidirectional Intra Coding for HEVC". In: *Proceedings of 32nd Picture Coding Symposium (PCS)*. Dez. 2016. DOI: 10.1109/PCS.2016.7906319.

[4] Thorsten Laude und Jörn Ostermann. „Deep learning-based intra prediction mode decision for HEVC". In: *Proceedings of 32nd Picture Coding Symposium (PCS)*. Dez. 2016. DOI: 10.1109/PCS.2016.7906399.

[5] Thorsten Laude, Yannick Richter und Jörn Ostermann. „Neural Network Compression using Transform Coding and Clustering". In: *NIPS Compact Deep Neural Network Representation with Industrial Applications Workshop*. Montreal, Canada, Dez. 2018. arXiv: 1805.07258.

[6] Thorsten Laude, Holger Meuel, Yiqun Liu und Jörn Ostermann. „Motion Blur Compensation in Scalable HEVC Hybrid Video Coding". In: *Proceedings of 30th Picture Coding Symposium*. Dez. 2013, S. 313–316. DOI: 10.1109/PCS.2013.6737746.

[7] Thorsten Laude, Xiaoyu Xiu, Jie Dong, Yuwen He, Yan Ye und Jörn Ostermann. „Improved Inter-Layer Prediction for the Scalable Extensions of HEVC". In: *Data Compression Conference (DCC)*. März 2014, S. 412. DOI: 10.1109/DCC.2014.45.

[8] Thorsten Laude, Xiaoyu Xiu, Jie Dong, Yuwen He, Yan Ye und Jörn Ostermann. „Scalable Extension of HEVC Using Enhanced Inter-Layer Prediction". In: *IEEE International Conference on Image Processing (ICIP)*. 2014, S. 3739–3743. DOI: 10.1109/ICIP.2014.7025759.

[9] Thorsten Laude, Yeremia Gunawan Adhisantoso, Jan Voges, Marco Munderloh und Jörn Ostermann. „A Comparison of JEM and AV1 with HEVC: Coding Tools, Coding Efficiency and Complexity". In: *Proceedings of the IEEE Picture Coding Symposium (PCS)*. Juni 2018.

[10] Bastian Wandt, Thorsten Laude, Yiqun Liu, Bodo Rosenhahn und Jörn Ostermann. „Extending HEVC Using Texture Synthesis". In: *IEEE Visual Communications and Image Processing (VCIP)*. Dez. 2017. DOI: 10.1109/VCIP.2017.8305034.

[11] Bastian Wandt, Thorsten Laude, Bodo Rosenhahn und Jörn Ostermann. „Detail-aware image decomposition for an HEVC-based texture synthesis framework". In: *Data Compression Conference (DCC)*. März 2018.

[12] Bastian Wandt, Thorsten Laude, Bodo Rosenhahn und Jörn Ostermann. „Extending HEVC with a Texture Synthesis Framework using Detail-aware Image Decomposition". In: *Proceedings of the Picture Coding Symposium (PCS)*. Juni 2018.

## PATENTE

[1] Thorsten Laude, Xiaoyu Xiu, Jie Dong, Yan Ye und Yuwen He. „Inter-layer Prediction for Scalable Video Coding“. In: *United States Patent US10148971B2* (Dez. 2018). URL: `https://patents.google.com/patent/US10148971B2`.

## STANDARDISIERUNGSBEITRÄGE

[1] R. Joshi, Y.-K. Wang, G. Tech, T. Laude, J. Chen, J. Xu, G. Sullivan, S Liu und Y. Ye. „JCTVC-V0031: Proposed editorial improvements to HEVC Screen Content Draft Text 4“. In: *Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 and ISO/IEC JTC1/SC29/WG11, 22th Meeting* (2015).

[2] Thorsten Laude. „JCTVC-R0113: Non-SCCE3: Improved Palette Index Coding with Contextualization“. In: *Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 and ISO/IEC JTC1/SC29/WG11, 18th Meeting* (2014).

[3] Thorsten Laude. „JCTVC-S0074: CE6: Results for Test B3 on Improved Palette Index Coding with Contextualization“. In: *Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 and ISO/IEC JTC1/SC29/WG11, 19th Meeting* (2014).

[4] Thorsten Laude. „JCTVC-S0075: Copy Mode for Static Screen Content“. In: *Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 and ISO/IEC JTC1/SC29/WG11, 19th Meeting* (2014).

[5] Thorsten Laude. „JCTVC-T0138: Copy mode for static screen content coding“. In: *Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 and ISO/IEC JTC1/SC29/WG11, 20th Meeting* (2015).

[6] Thorsten Laude. „JCTVC-U0119: On slice segment freeze signalling for screen content coding“. In: *Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 and ISO/IEC JTC1/SC29/WG11, 21th Meeting* (2015).

[7] Thorsten Laude. „JCTVC-V0032: Minor editorial improvements for HEVC SCC.“ In: *Joint Collaborative Team on Video Coding(JCT-VC) of ITU-T SG16and ISO/IEC JTC1/SC29/WG11, 22th Meeting*. Okt. 2015.

## PATENTANTRÄGE

[1] Erfinder: Thorsten Laude, Felix Haub und Jörn Ostermann. *Inter Coding using Deep Recurrent Neural Networks and Artificial Reference Pictures (62/772,100, US, anhängig)*. 2019.

[2] Erfinder: Thorsten Laude, Florian Mehringer und Jörn Ostermann. Anmelder und Erfinder: Zhijie Zhao. *Encoder- and Decoder-side Device for Machine Learning-based Signal Coding (PCT/EP2019/052342, anhängig)*. 2019.

[3] Erfinder: Thorsten Laude, Marco Munderloh und Jörn Ostermann. Anmelder und Erfinder: Zhijie Zhao. *Apparatus and Method for Encoding an Image (PCT/EP2016/079663, anhängig)*. 2016.

[4] Erfinder: Thorsten Laude, Marco Munderloh und Jörn Ostermann. *Improved Screen Content and Mixed Content Coding (PCT/IB2015/051821, anhängig)*. 2015.

[5] Erfinder: Thorsten Laude, Jörn Ostermann, Marco Munderloh und Haoping Yu. *Improved Method for Screen Content Coding (PCT/US2015/020505, anhängig)*. 2015.

[6] Erfinder: Thorsten Laude, Stella Grasshof, Marco Munderloh und Jörn Ostermann. *Devices and Methods for Video Coding Using Intra Prediction (PCT/EP2016/066988, anhängig)*. 2016.

[7] Erfinder: Bastian Wandt, Yiqun Liu, Thorsten Laude, Jörn Ostermann und Bodo Rosenhahn. Anmelder und Erfinder: Zhijie Zhao. *Frequency Adjustment for Texture Synthesis in Video Coding (PCT/EP2017/082071, anhängig)*. 2017.

[8] Erfinder: Bastian Wandt, Yiqun Liu, Thorsten Laude, Jörn Ostermann und Bodo Rosenhahn. Anmelder und Erfinder: Zhijie Zhao. *Polynomial Fitting for Motion Compensation and Luminance Reconstruction in Texture Synthesis (PCT/EP2017/082072, anhängig)*. 2017.

[9] Erfinder: Bastian Wandt, Yiqun Liu, Thorsten Laude, Jörn Ostermann und Bodo Rosenhahn. Anmelder und Erfinder: Zhijie Zhao. *Cluster Refinement for Texture Synthesis in Video Coding (PCT/EP2018/057477, anhängig)*. 2018.

# Thorsten Laude

## Persönliche Daten

| | |
|---|---|
| Geburtsjahr: | 1988 |
| Geburtsort: | Hannover |
| E-Mail: | thorstenlaude@mailbox.org |

## Beruflicher Werdegang

| | |
|---|---|
| Seit 2020 | Landeskriminalamt Niedersachsen |
| 2013 – 2020 | Wissenschaftlicher Mitarbeiter,<br>Institut für Informationsverarbeitung,<br>Leibniz Universität Hannover,<br>Betreuer: Prof. Dr.-Ing. Jörn Ostermann |
| 2013 | Interdigital Communications, San Diego, USA |
| 2012 | Studentische Hilfskraft,<br>Institut für Informationsverarbeitung,<br>Leibniz Universität Hannover |
| 2010 – 2012 | Studentische Hilfskraft,<br>Institut für Kommunikationstechnik,<br>Leibniz Universität Hannover |

## Ausbildung

| | |
|---|---|
| 2008 – 2013 | Studium der Elektrotechnik,<br>Leibniz Universität Hannover,<br>Spezialisierung: Nachrichtentechnik,<br>Abschluss: Diplom-Ingenieur |
| 2008 | Abitur, Gymnasium Langenhagen |

## Die Reihen der Fortschritt-Berichte VDI:

1 Konstruktionstechnik/Maschinenelemente
2 Fertigungstechnik
3 Verfahrenstechnik
4 Bauingenieurwesen
5 Grund- und Werkstoffe/Kunststoffe
6 Energietechnik
7 Strömungstechnik
8 Mess-, Steuerungs- und Regelungstechnik
9 Elektronik/Mikro- und Nanotechnik
10 Informatik/Kommunikation
11 Schwingungstechnik
12 Verkehrstechnik/Fahrzeugtechnik
13 Fördertechnik/Logistik
14 Landtechnik/Lebensmitteltechnik
15 Umwelttechnik
16 Technik und Wirtschaft
17 Biotechnik/Medizintechnik
18 Mechanik/Bruchmechanik
19 Wärmetechnik/Kältetechnik
20 Rechnerunterstützte Verfahren (CAD, CAM, CAE CAQ, CIM ...)
21 Elektrotechnik
22 Mensch-Maschine-Systeme
23 Technische Gebäudeausrüstung

ISBN 978-3-18-387110-0