Johannes Anastasiadis* und Fernando Puente León

Detektion von Stoffen in Lebensmitteln mit Hilfe von 3D-Faltungsautoencodern

Detection of substances in food with 3D convolutional autoencoders

DOI 10.1515/teme-2018-0033

Zusammenfassung: Hyperspektrale Bilder beinhalten dank ihrer hohen spektralen Auflösung wertvolle Informationen. Sie können beispielsweise zur berührungslosen Untersuchung von Lebensmitteln verwendet werden. Um diese mit Hilfe konvolutionaler neuronaler Netze verarbeiten zu können, werden jedoch große Lernstichproben benötigt. Dies gilt insbesondere, wenn die Daten nicht vorverarbeitet werden und deshalb eine hohe Dimension besitzen. Allerdings existieren nur verhältnismäßig wenige hyperspektrale Datensätze. Um dieses Problem zu umgehen, kann ein Vortraining mit einem Autoencoder durchgeführt werden. Dieser komprimiert das Bild und rekonstruiert es im Anschluss wieder. Der Autoencoder wird trainiert, indem der Rekonstruktionsfehler minimiert wird. Bei der Komprimierung entstehen so aussagekräftige Merkmale. In diesem Beitrag wird am Beispiel von Gewürzmischungen untersucht, ob einzelne Komponenten in Mischungen detektiert werden können. Dabei wird mit einem kleinen Datensatz ein neuronales Netz mit Hilfe eines 3D-Faltungsautoencoders trainiert.

Schlüsselwörter: Autoencoder, Hyperspektralbild, neuronale Netze, dreidimensionale Faltung.

Abstract: Hyperspectral images contain very useful information because of their high spectral resolution, which can be used for non-contact food testing. However, to process them with convolutional neural networks, large data sets are needed. This is especially true if the data is not preprocessed and therefore of high dimension. However, relatively few hyperspectral data sets exist. To solve this problem, the neural network can be pre-trained using an autoencoder, which compresses and reconstructs the image. By minimizing the reconstruction error, useful features can be learned to solve the original task. In this work, spice mixtures are used to investigate whether individual components can be detected. In particular, a neural network using a 3D convolutional autoencoder is trained with a small data set.

Keywords: Autoencoder, hyperspectral image, neural networks, three-dimensional convolution.

***Korrespondenzautor: Johannes Anastasiadis,** Karlsruher Institut für Technologie, Institut für Industrielle Informationstechnik (IIIT), E-Mail: anastasiadis@kit.edu

Fernando Puente León, Karlsruher Institut für Technologie, Institut für Industrielle Informationstechnik (IIIT)

1 Einleitung

Optische Messverfahren spielen bei der Untersuchung von Lebensmitteln eine große Rolle, da sie berührungslos und nicht-destruktiv sind. Einsatzgebiete sind beispielsweise die Detektion unerwünschter Stoffe in Lebensmitteln und die Bestimmung der Mengenanteile ihrer Komponenten. In vielen Fällen genügen die drei Farbkanäle (Rot, Grün, Blau), die im menschlichen Auge und bei Farbkameras verwendet werden, nicht, um diese Aufgaben zu lösen. An dieser Stelle kommen hyperspektrale Aufnahmen mit bis zu mehreren hundert schmalbandigen Wellenlängenkanälen zum Einsatz [5]. Hieraus können mit geeigneten Methoden Rückschlüsse auf die Materialeigenschaften gezogen werden [1, 8–10].

Künstliche neuronale Netze konnten in den vergangenen Jahren große Erfolge erzielen. Dazu gehört die Klassifikation von Bildern, zu der sich vor allem konvolutionale neuronale Netze (KNN) eignen [11]. Auch für hyperspektrale Bilder werden bereits KNN verwendet. Dabei werden jedoch hauptsächlich einzelne Pixelspektren betrachtet [6] oder es findet, wie bei Farbbildern, nur eine Faltung in die örtlichen Richtungen statt [13, 18]. Andere Ansätze versuchen, die örtliche und die spektrale Information getrennt zu extrahieren und später miteinander zu verknüpfen [2, 3].

Hyperspektrale Bilder haben den Nachteil, dass nur kleine Lernstichproben, die zum Training der neuronalen Netze benötigt werden, existieren. Dieses Problem wird dadurch verstärkt, dass aufgrund der hohen Anzahl an Wellenlängenkanälen die Anzahl der zu lernenden Parameter groß ist. Da es sich beim Training eines künstlichen neuronalen Netzes im Allgemeinen um ein nicht konvexes Optimierungsproblem handelt, hängt der Erfolg des Trainingsvorgangs von den Startwerten der Parameter

ab. Geeignete Startparameter können mit einem unüberwachten Vortraining gefunden werden [4]. Eine mögliche Umsetzung des unüberwachten Trainings ist der Autoencoder. Für eine Klassifikation von Farbbildern mit drei Farbkanälen wurden in [14] erfolgreich Autoencoder eingesetzt. In [3] werden Autoencoder für hyperspektrale Bilder im Bereich der Fernerkundung verwendet.

Im Rahmen dieses Beitrags soll der Einsatz eines KNN zur Detektion von Stoffen in Lebensmitteln untersucht werden. Dabei wird in den Faltungsschichten eine dreidimensionale Faltung durchgeführt, um die örtliche mit der spektralen Information zu verknüpfen. In [12] wurden bereits KNN mit einer dreidimensionalen Faltung für hyperspektrale Bilder zur Durchführung einer Klassifikation angewendet. In diesem Beitrag liegt der Schwerpunkt auf der Untersuchung eines Autoencoders in Verbindung mit einem solchen Netz und dessen Auswirkungen auf das Trainingsverhalten.

Der Beitrag gliedert sich wie folgt: In Abschnitt 2 werden die Grundlagen zu hyperspektralen Bildern und neuronalen Netzen vorgestellt. Anschließend wird in Abschnitt 3 die Architektur des verwendeten 3D-Faltungsautoencoders präsentiert. Eine Auswertung der Ergebnisse erfolgt in Abschnitt 4. Hierzu wird zunächst die Rekonstruktionsfähigkeit des Autoencoders und im Anschluss daran die Auswirkung des Vortrainings auf die Detektion untersucht. Zuletzt wird in Abschnitt 5 eine kurze Zusammenfassung gegeben, bevor mögliche zukünftige Forschungsgegenstände aufgeführt werden.

2 Grundlagen

In diesem Abschnitt wird zunächst auf den Aufbau hyperspektraler Bilder und die verwendete dreidimensionale Faltung eingegangen. Im Anschluss folgt eine kurze Einführung in neuronale Netze als Basis für die KNN sowie die Darstellung der allgemeinen Struktur eines darauf basierenden Faltungsautoencoders.

2.1 Hyperspektrale Bilder

Hyperspektrale Bilder besitzen bis zu mehrere hundert schmalbandige Wellenlängenkanäle. Die Wellenlängen reichen, je nach Anwendung, vom ultravioletten Bereich bis in den langwelligen Infrarotbereich des elektromagnetischen Spektrums. Hyperspektrale Bilder haben ihren Ursprung in der Fernerkundung. Dank schneller Aufnahmeverfahren und Datenverarbeitung kommen hyperspektrale Bilder immer häufiger in industriellen Anwendungen zum Einsatz.

Mathematisch können hyperspektrale Bilder als dreidimensionale Datenwürfel betrachtet werden. Dabei gibt es zwei örtliche Koordinaten und eine in Richtung der Wellenlänge. Jedem Punkt dieses Würfels wird ein Wert zugeordnet, der als Intensitätswert für das entsprechende Band betrachtet werden kann. Die dreidimensionalen Datenwürfel können mit einem 3D-Faltungskern gefaltet werden. Dies geschieht analog zur eindimensionalen Faltung für alle Dimensionen [12].

2.2 Neuronale Netze

Neuronale Netze bestehen aus Neuronen. Jedes Neuron hat mehrere Eingänge und einen Ausgang. Die Eingänge werden mit Kantengewichten multipliziert, anschließend wird ein Schwellenwert addiert. Das Resultat wird zuletzt durch eine nichtlineare Funktion modifiziert. Die Neuronen sind in Schichten angeordnet, wobei jeder Ausgang der vorherigen Schicht mit allen Neuronen der nachfolgenden Schicht verbunden ist. Verbindungen innerhalb einer Schicht sind in *Feed-forward*-Netzen, wie sie hier verwendet werden, nicht vorgesehen. Eine ganze Schicht lässt sich analog zu einem Neuron mathematisch durch

$$\mathbf{h} = \boldsymbol{\varphi}(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{1}$$

beschreiben. Dabei ist $\mathbf{h} \in \mathbb{R}^K$ der Ausgangsvektor und $\mathbf{x} \in \mathbb{R}^J$ der Eingangsvektor der Schicht, $\mathbf{W} \in \mathbb{R}^{K \times J}$ sind die Kantengewichte und $\mathbf{b} \in \mathbb{R}^K$ die Schwellenwerte. Die nichtlineare Funktion $\boldsymbol{\varphi}$ wird elementweise angewandt.

Das neuronale Netz wird trainiert, indem mit Hilfe von bekannten Ein- und Ausgangsgrößen die Parameter ($\mathbf{W}$ und $\mathbf{b}$ aller Schichten) schrittweise durch Minimierung einer Kostenfunktion angepasst werden. Mit Hilfe von *backpropagation* können so die Änderungen der Parameter berechnet werden [16].

2.3 Konvolutionale neuronale Netze

Bei KNN wird die Multiplikation in (1) durch eine Faltung ersetzt, was mehrere Vorteile mit sich bringt. Da nicht alle Neuronen miteinander verbunden sind, werden wesentlich weniger Parameter benötigt. Die Faltungskerne sind an jeder Stelle der Daten gleich, was sowohl zu weniger benötigten Parametern als auch zu einer Ortsinvarianz bezüglich der Merkmale führt. Bei mehrdimensionalen Daten wird ihre Anordnung relativ zueinander berücksich-

tigt. Dadurch sind KNN für die Bearbeitung von Bildern besonders geeignet.

Neben den Faltungsschichten besitzen KNN in den meisten Fällen auch *Pooling*-Schichten. Diese fassen benachbarte Werte in einem einzelnen Wert zusammen. Dazu wird meist das Maximum oder der Mittelwert der Nachbarschaft herangezogen. Mit Hilfe der *Pooling*-Schichten kann erreicht werden, dass redundante Information verworfen wird. Damit wird versucht, dem *Overfitting* – dem zu starken Anpassen der Parameter an den verwendeten Trainingsdatensatz – entgegenzuwirken. Außerdem wird eine Ortsinvarianz innerhalb der Nachbarschaft erreicht.

2.4 Faltungsautoencoder

Autoencoder sind neuronale Netze, die unüberwacht trainiert werden. Dazu werden die Eingangsdaten in einem Teil, dem *Encoder*, verarbeitet. Bei einem Faltungsautoencoder ist dies ein KNN. Der zweite Teil, der *Decoder*, rekonstruiert die Eingangsdaten (siehe Abb. 1). Als Kostenfunktion wird der Fehler zwischen Original und Rekonstruktion verwendet [15]. Die Dimension der Daten am Ausgang des *Encoders* ist meist geringer als die der Eingangsdaten. Dadurch können im *Encoder* relevante Merkmale erlernt werden. Nach erfolgreichem Training wird der *Decoder* verworfen. Die extrahierten Merkmale können beispielsweise für eine Klassifikationsaufgabe herangezogen werden.

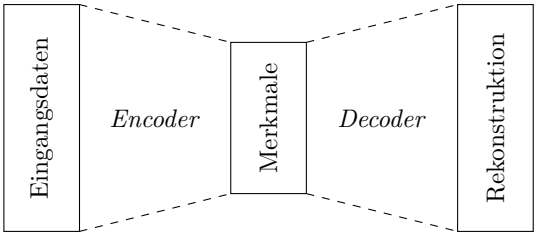

Abb. 1: Struktur eines Autoencoders.

Werden bei einem Faltungsautoencoder im *Encoder* *Pooling*-Schichten verwendet, so wird auch eine inverse Schicht, eine *Unpooling*-Schicht, im *Decoder* benötigt. Eine Möglichkeit ist die Erhöhung der Auflösung mit Nullen und die Beibehaltung der Position des ursprünglichen Maximums [19]. Dadurch werden Fehler durch Verschiebung vermieden. In Abb. 2 wird der Wert 7,8 beim *Unpooling* an die Stelle gesetzt, an der beim *Pooling* das Maximum mit dem Wert 9 zu finden ist.

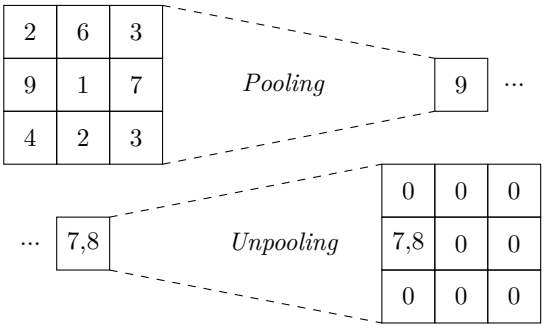

Abb. 2: Beispiel für *Unpooling* mit Festhalten der Position.

3 Verwendete Netzstruktur

Das in diesem Beitrag verwendete neuronale Netz lässt sich in zwei Teile gliedern. Zum einen in den konvolutionalen Teil, zum anderen in den Detektor. Der konvolutionale Teil hat die Aufgabe, Merkmale zu extrahieren. Er wird in einem Vortraining als Autoencoder realisiert. Der Detektor hat wiederum die Aufgabe, die einzelnen Stoffe zu detektieren und wird für jeden Stoff, der getestet wird, trainiert.

3.1 3D-Faltungsautoencoder

Das KNN besteht aus sechs Schichten. Eine Faltungsschicht lässt sich mit dem 4-Tupel $(w_\mathrm{f}, w_\mathrm{x}, w_\mathrm{y}, w_\lambda)$ beschreiben. Dabei beschreibt w_f die Anzahl der Merkmale, mit denen der Ausgang der vorherigen Schicht gefaltet wird, während w_x, w_y und w_λ die Größe der Faltungskerne in der jeweiligen Dimension beischreiben.

Als Aktivierungsfunktion wird in diesem Beitrag die Sigmoidfunktion $\sigma : \mathbb{R} \to \mathbb{R}$ verwendet mit:

$$\sigma(x) = \frac{1}{1 + \mathrm{e}^{-x}} \,. \tag{2}$$

Die *Pooling*-Schicht lässt sich mit Hilfe des 3-Tupels $(p_\mathrm{x}, p_\mathrm{y}, p_\lambda)$ beschreiben. Dabei sind $p_\mathrm{x}, p_\mathrm{y}, p_\lambda \in \mathbb{N}^+$ die Ausdehnung der zum *Pooling* herangezogenen Nachbarschaft in die jeweilige Richtung. In diesem Beitrag wird Maximum-*Pooling* ohne Überlappung verwendet.

Die Entfaltung im *Decoder* wird, wie in [14], mit einer Faltung realisiert. Allerdings werden im Gegensatz zu [14] eigene Parameter für den *Decoder* verwendet. Entfaltungsschichten können mit dem gleichen 4-Tupel beschrieben werden, mit dem auch die Faltung beschrieben wird. Das *Unpooling* erfolgt nach Abb. 2. Es lässt sich mit einem 3-Tupel analog zu dem 3-Tupel für *Pooling* beschreiben.

Der schichtweise Aufbau des 3D-Faltungsautoencoders ist in Abb. 3 zu sehen. Beim Training wird der mittle-

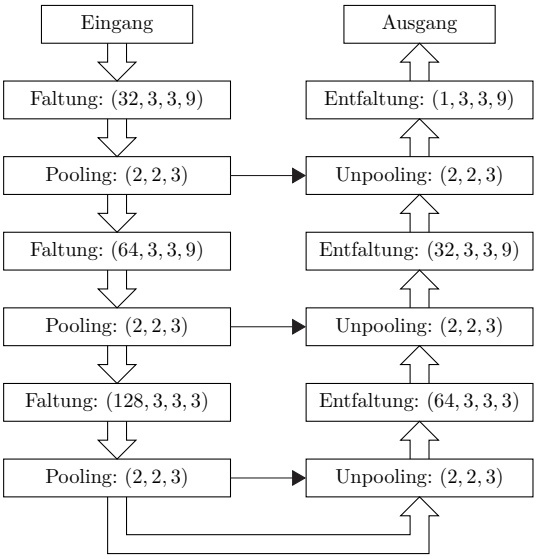

Abb. 3: Aufbau des verwendeten 3D-Faltungsautoencoders. Die Pfeile übertragen vom *Encoder* (linke Spalte) zum *Decoder* (rechte Spalte) die Information, an welcher Stelle sich das Maximum einer Nachbarschaft befindet.

re quadratische Fehler (MSE) zwischen Originalbild und Rekonstruktion minimiert.

3.2 Detektor

Der Detektor besteht aus dem vortrainierten *Encoder*, der mit einem neuronalen Netz bestehend aus vier Schichten verbunden wird. Die erste und die dritte Schicht sind vollständig verbundene Schichten (VVS). Dazwischen befindet sich eine *Dropout*-Schicht. Sie setzt beim Training zufällig einen festen Anteil der Ausgänge der ersten Schicht zu null. Dadurch wird die Gefahr von *Overfitting* verringert [17]. Als Ausgangsschicht dient eine *Softmax*-Schicht $\xi : \mathbb{R}^K \to \mathbb{R}^K$. Sie sorgt dafür, dass die Ausgänge in der Summe eins ergeben und positiv sind, sodass sie als Wahrscheinlichkeiten

$$\xi(\mathbf{x}) = \frac{\mathrm{e}^{x_\kappa}}{\sum_{\tilde{\kappa}=1}^{K} \mathrm{e}^{x_{\tilde{\kappa}}}} \quad \text{für} \quad \kappa = 1, 2, \ldots, K \tag{3}$$

interpretiert werden können. Die erste und die dritte Schicht lassen sich jeweils durch die Anzahl ihrer Ausgänge $\mathrm{w_o}$ beschreiben. Die Eingänge werden durch die Vorgängerschichten bestimmt. Die *Dropout*-Schicht kann durch den Anteil $\mathrm{d_0} \in \mathbb{R} \mid 0 \leq \mathrm{d_0} \leq 1$ der Verbindungen beschrieben werden, die zu null gesetzt werden (siehe Abb. 4).

Das Training erfolgt hier durch Minimierung der Kreuzentropie zwischen den Ausgangswerten und den Sollwerten der Trainingsdaten. Diese bestehen immer aus einer Eins und einer Null. Die Anordnung hängt davon ab, ob in der untersuchten Mischung der Stoff enthalten ist oder nicht.

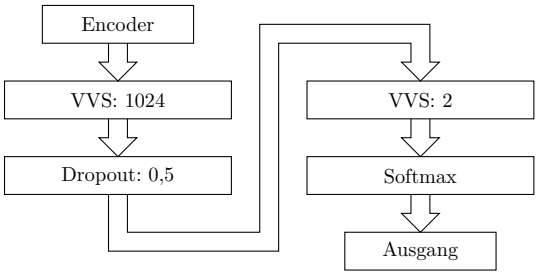

Abb. 4: Struktur des Detektors. Die Aktivierungsfunktion nach der ersten Schicht ist die Sigmoidfunktion (2); nach der dritten Schicht wird keine Nichtlinearität angewandt. Die beiden Ausgangssignale geben an, ob eine Detektion erfolgt oder nicht, abhängig davon, welcher Wert größer ist.

4 Ergebnisse

Um die in Abschnitt 3 beschriebenen Netze zu trainieren und zu testen, wurde ein Datensatz verwendet, der im eigenen Bildverarbeitungslabor aufgenommen wurde. Der Datensatz besteht aus 296 hyperspektralen Bildern von gemahlenen Gewürzmischungen und enthält Mischungen bestehend aus einer, zwei, drei oder vier Komponenten. Es gibt insgesamt elf Komponenten (siehe Abb. 7).

Jedes hyperspektrale Bild hat eine örtliche Auflösung von 24×24 Pixeln und wurde in 91 Wellenlängen, von 450 nm bis 810 nm, abgetastet. Die Bilder sind in einen Trainingsdatensatz mit 233 Bildern und einen Testdatensatz mit 63 Bildern aufgeteilt.

4.1 Rekonstruktionsfähigkeit des 3D-Faltungsautoencoders

Das Gütemaß zur Darstellung der Rekonstruktionsfähigkeit $\gamma \in \mathbb{R}^+$ ist das Verhältnis der mittleren Pixelenergie zur mittleren Fehlerenergie.

Da die Optimierung des 3D-Faltungsautoencoders kein konvexes Optimierungsproblem darstellt, wurde mit verschiedenen zufälligen Startparametern sowie verschiedenen Optimierern und Optimierungsparametern trainiert. Die besten Ergebnisse konnten mit dem *adam*-Optimierer [7] erzielt werden. Die Optimierung erfolgte nach dem *Minibatch*-Verfahren, d. h. in den Trainingsiterationen werden mehrere (hier: 100), aber nicht alle Bilder für die aktuelle Iteration herangezogen.

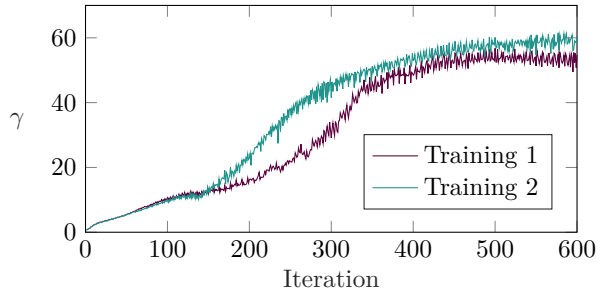

Abb. 5: Gütemaß γ während des Trainings des Autoencoders.

In Abb. 5 ist das Gütemaß γ für zwei Trainingsverläufe zu je 600 Iterationen zu sehen. Der höchste Wert $\gamma = 60{,}1$ beim Test wurde nach Trainingsvorgang 1 erzielt. Trainingsvorgang 2 erreichte den besten Wert für γ während des Trainings, konnte beim Testen jedoch mit $\gamma = 58{,}4$ nur den zweitbesten Wert erzielen. Die Kurven erscheinen verrauscht, weil die *Dropout*-Schicht dazu führt, dass in jedem Schritt das Netz stark verändert wird.

Es zeigt sich, dass der 3D-Faltungsautoencoder zumeist durchaus in der Lage ist, Bilder zu rekonstruieren. Das gelingt, obwohl nach der dritten *Pooling*-Schicht das Bild auf 6,6 % seiner ursprünglichen Größe komprimiert wird. Somit liegt es nahe, dass der *Encoder* nach dem Training signifikante Merkmale extrahieren kann.

Der 3D-Faltungsautoencoder kann auch schichtweise trainiert werden [14]. Dazu werden zunächst die beiden äußeren Schichten trainiert, indem die erste und die letzte *Pooling*-Schicht direkt miteinander verbunden werden. Die nächste Schicht wird dann trainiert, indem die zweite und die vorletzte *Pooling*-Schicht miteinander verbunden werden. Allerdings werden die Parameter der ersten Schicht konstant gehalten und der MSE wird zwischen dem Ausgang der ersten und dem der vorletzten Schicht berechnet. So wird weiter verfahren, bis alle Schichten trainiert sind. Welche Schicht wann trainiert wird, wird in Abb. 6 veranschaulicht. Dazu werden die Faltungsschichten aus Abb. 3 von eins bis sechs durchnummeriert.

Trainiert und getestet wurde mit dem Datensatz aus Abschnitt 4. Das beste erreichte Ergebnis für die Rekonstruktion war $\gamma = 17{,}7$. Somit eignet sich für eine Komprimierung und eine Rekonstruktion das simultane Training aller Schichten besser.

1. Durchgang:	2. Durchgang:	3. Durchgang:
1, 6	1, **2, 5**	1, 2, **3, 4**

Abb. 6: Beteiligung der Schichten an den Trainingsdurchgängen. Fett geschriebene Ziffern stehen für Schichten, deren Parameter trainiert werden. Die restlichen Schichten werden konstant gehalten.

Tab. 1: Gütemaß γ, bezogen auf die Testdaten, für alle verwendeten Autoencoder (*schichtweise trainiert).

	AE-V	AE-V*	AE-R	AE-R*
γ (Testdatensatz)	34,6	14,3	39,1	14,0

4.2 Detektion

In diesem Abschnitt wird untersucht, wie gut das Netz, bestehend aus *Encoder* und Detektor, mit unterschiedlichen Startparametern trainiert wird. Dazu wird unterschieden, ob und auf welche Art und Weise ein Vortraining stattgefunden hat. Zum Vergleich der Methoden wird der Anteil der korrekten Entscheidungen ζ im Testdatensatz herangezogen. Dieser Wert wird für verschiedene Anzahlen an Trainingsiterationen τ, die vor dem Testen stattgefunden haben, berechnet. In jeder Trainingsiteration wird der vollständige Trainingsdatensatz herangezogen. Zur besseren Vergleichbarkeit sind alle Optimierungsparameter für alle Methoden gleich.

In Abb. 7 sind diese Ergebnisse für alle elf Gewürze dargestellt. Zu Gunsten der Lesbarkeit sind die Kurven mit Hilfe eines *Moving-Average*-Filters geglättet worden. Für das Vortraining wurde der Autoencoder einmal mit dem vollständigen Trainingsdatensatz (AE-V) und einmal mit den Trainingsdaten (AE-R), die nur die Reinstoffe enthalten, trainiert. Beide Autoencoder wurden jeweils simultan und schichtweise trainiert (siehe Tabelle 1).

Dabei zeigt sich, dass für die Reproduktion der reduzierte Trainingsdatensatz, der nur die Reinstoffe beinhaltet, völlig ausreicht. Für alle Gewürze gilt, dass das Vortraining nötig ist, um gute Detektionsergebnisse zu erhalten. In Abb. 7 wird dies durch eine konstante Rate für richtige Entscheidungen sichtbar. Sie scheint zunächst zwar relativ hoch zu sein, was aber daran liegt, dass die Ausgangswerte nicht gleichverteilt sind. Es sind immer weniger Stichproben ohne als mit dem entsprechenden Gewürz vorhanden. Die konstante Rate entsteht, wenn der Detektor immer für alle Proben gleich entscheidet und somit nicht verwendbar ist.

Wird ein Vortraining durchgeführt, so verbessert sich das Ergebnis erheblich. Nur für Koriander zeigt sich kaum eine Verbesserung der Testergebnisse. Für Kurkuma, Kreuzkümmel, Paprika und Cayennepfeffer kann, nach einer gewissen Anzahl an Trainingsdurchgängen, eine nahezu perfekte Trefferquote erzielt werden.

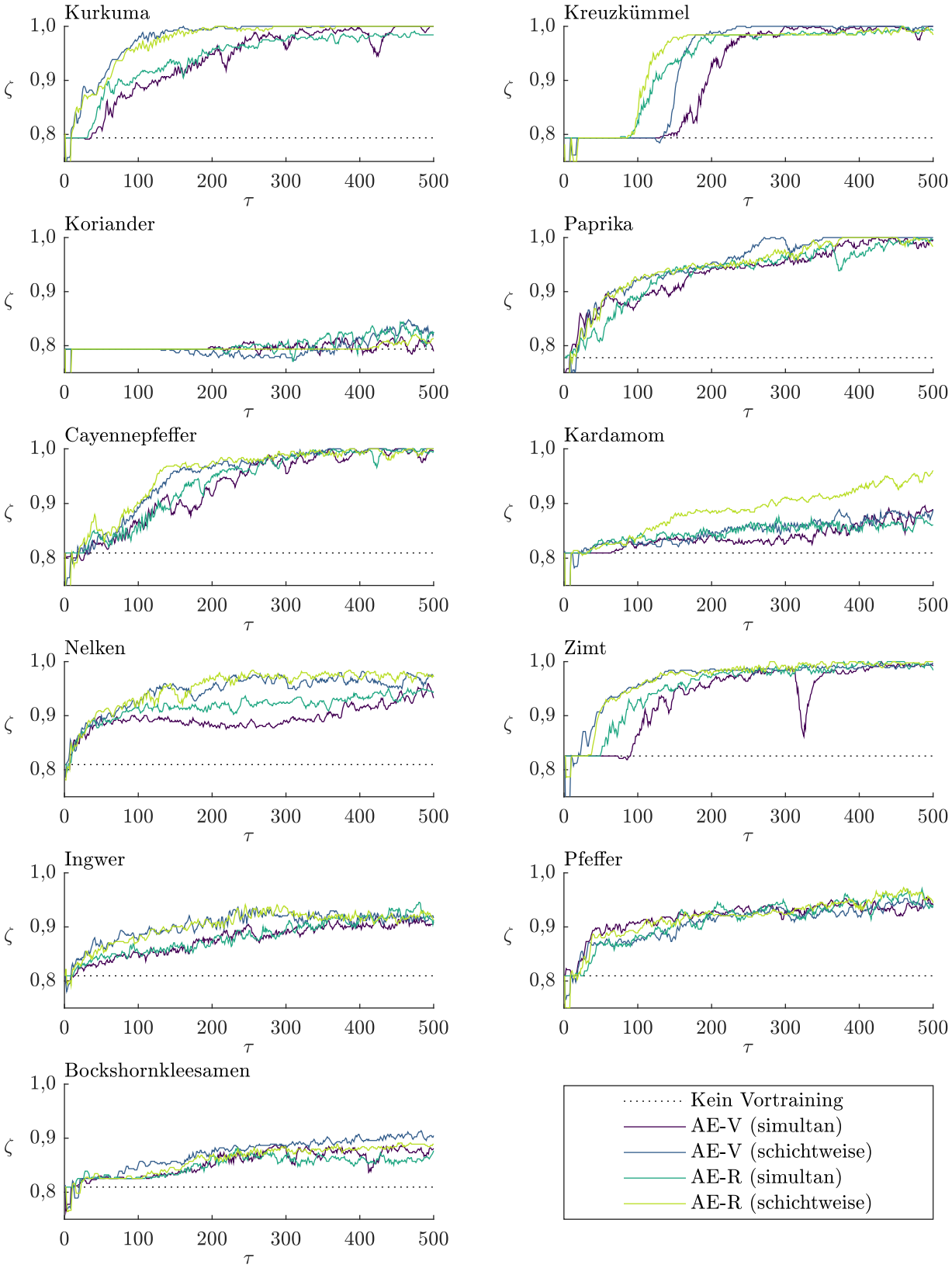

Abb. 7: Anteil der richtigen Entscheidungen beim Testdatensatz für alle Gewürze. Dabei wird zwischen den verwendeten Vortrainingsmethoden verglichen. Wünschenswert ist dabei eine schnelle Konvergenz gegen eins.

Des Weiteren fällt auf, dass die schichtweise vortrainierten Parameter meist zu einem besseren Detektionsergebnis führen, obwohl diese einen größeren Rekonstruktionsfehler beim Autoencoder besitzen (vgl. Tabelle 1). Dies könnte damit zu erklären sein, dass es bei einem simultanen Vortraining mehr freie Parameter gibt. Dadurch steigt die Gefahr, dass sich die Parameter zu stark an den Trainingsdatensatz anpassen bzw. sich zu stark an die Reproduktionsaufgabe anpassen.

Das Vortraining nur mit den Reinstoffen durchzuführen führt nur zu leichten Veränderungen, die zum Teil positiv und zum Teil negativ ausfallen. Dies zeigt, dass wenig Daten für das Vortraining genügen und trotzdem teils bessere Ergebnisse erzielt werden können.

5 Zusammenfassung und Ausblick

Das Training neuronaler Netze erfordert, vor allem für hochdimensionale hyperspektrale Bilder, große Lernstichproben. Da diese in den meisten Fällen nicht verfügbar sind, ist ein geschicktes Vortraining von Nöten, welches die Parameter so einstellt, dass damit bereits aussagekräftige Merkmale extrahiert werden können. Die Ergebnisse zeigen, dass ein Detektor mit Hilfe des Vortrainings, trotz der großen unverarbeiteten dreidimensionalen Datenwürfel und kleiner Lernstichprobe, erfolgreich trainiert werden kann. Dabei hat ein schichtweises Vortraining häufig zu besseren Trefferquoten geführt. Außerdem genügt es, dies mit einem eingeschränkten Datensatz, der nur die hyperspektralen Bilder der Reinstoffe enthält, durchzuführen. Dadurch kann *Overfitting* beim Vortraining verhindert werden.

Die Ergebnisse zeigen auch, dass nicht für alle Komponenten erfolgreich ein Detektor trainiert werden kann. Hier gilt es, zu prüfen, ob Hyperspektralbilder mit einem größeren Wellenlängenbereich zu einem besseren Ergebnis führen. Des Weiteren sollte zukünftig die Auswirkung des Vortrainings auf weitere, vor allem tiefere, Netzstrukturen untersucht werden.

Literatur

[1] S. Bauer, J. Stefan und F. Puente León. Hyperspectral image unmixing involving spatial information by extending the alternating least-squares algorithm. *tm – Technisches Messen*, 82(4):174–186, 2015.

[2] C. Chen, F. Jiang, C. Yang, S. Rho, W. Shen, S. Liu und Z. Liu. Hyperspectral classification based on spectral-spatial convolutional neural networks. *Engineering Applications of Artificial Intelligence*, 68:165–171, 2018.

[3] Y. Chen, Z. Lin, X. Zhao, G. Wang und Y. Gu. Deep learning-based classification of hyperspectral data. *IEEE Journal of Selected topics in applied earth observations and remote sensing*, 7(6):2094–2107, 2014.

[4] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent und S. Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11 (Feb):625–660, 2010.

[5] A. Gowen, C. O'Donnell, P. Cullen, G. Downey und J. Frias. Hyperspectral imaging – an emerging process analytical tool for food quality and safety control. *Trends in Food Science & Technology*, 18(12):590–598, 2007.

[6] W. Hu, Y. Huang, L. Wei, F. Zhang und H. Li. Deep convolutional neural networks for hyperspectral image classification. *Journal of Sensors*, 2015.

[7] D. Kingma und J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[8] W. Krippner und F. Puente León. Optische Bandselektion für die verbesserte Schätzung von Materialanteilen. *tm – Technisches Messen*, 84(S1):S124–S130, 2017.

[9] W. Krippner, S. Bauer und F. Puente León. Ortsaufgelöste optische Bestimmung von Materialanteilen in Mischungen. *tm – Technisches Messen*, 84(3):207–215, 2017.

[10] W. Krippner, S. Bauer und F. Puente León. Considering spectral variability for optical material abundance estimation. *tm – Technisches Messen*, 85(3):149–158, 2018.

[11] Y. LeCun, Y. Bengio et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[12] Y. Li, H. Zhang und Q. Shen. Spectral-spatial classification of hyperspectral imagery with 3d convolutional neural network. *Remote Sensing*, 9(1):67, 2017.

[13] K. Makantasis, K. Karantzalos, A. Doulamis und N. Doulamis. Deep supervised learning for hyperspectral data classification through convolutional neural networks. In *Geoscience and Remote Sensing Symposium (IGARSS), 2015 IEEE International*, S. 4959–4962. IEEE, 2015.

[14] J. Masci, U. Meier, D. Cireşan und J. Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. *Artificial Neural Networks and Machine Learning–ICANN 2011*, S. 52–59, 2011.

[15] M. Ranzato, F. J. Huang, Y. L. Boureau und Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *IEEE Conf. on Computer Vision and Pattern Recognition*, S. 1–8, June 2007.

[16] D. E. Rumelhart, G. E. Hinton und R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323 (6088):533, 1986.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever und R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[18] S. Yu, S. Jia und C. Xu. Convolutional neural networks for hyperspectral image classification. *Neurocomputing*, 219: 88–98, 2017.

[19] J. Zhao, M. Mathieu, R. Goroshin und Y. LeCun. Stacked what-where auto-encoders. *arXiv preprint arXiv:1506.02351*, 2016.