Knuth, Nico; Nastansky, Andreas

Working Paper

Anwendung von Deep Learning in der Prognose der Volatilität des DAX: Ein Vergleich der Prognosegüte von GARCH und LSTM

Statistische Diskussionsbeiträge, No. 59

Provided in Cooperation with:
Wirtschafts- und Sozialwissenschaftliche Fakultät, Universität Potsdam

Suggested Citation: Knuth, Nico; Nastansky, Andreas (2025) : Anwendung von Deep Learning in der Prognose der Volatilität des DAX: Ein Vergleich der Prognosegüte von GARCH und LSTM, Statistische Diskussionsbeiträge, No. 59, Universität Potsdam, Potsdam, https://doi.org/10.25932/publishup-67486

This Version is available at:
https://hdl.handle.net/10419/337367

UNIVERSITÄT POTSDAM

Andreas Nastansky (Hrsg.)

STATISTISCHE DISKUSSIONSBEITRÄGE

Nr. 59

Nico Knuth Andreas Nastansky

Anwendung von Deep Learning in der Prognose der Volatilität des DAX: Ein Vergleich der Prognosegüte von GARCH und LSTM

Potsdam 2025

ISSN 0949-068X

STATISTISCHE DISKUSSIONSBEITRÄGE

Nr. 59

Nico Knuth Andreas Nastansky

Anwendung von Deep Learning in der Prognose der Volatilität des DAX: Ein Vergleich der Prognosegüte von GARCH und LSTM

Autoren: Nico Knuth, B.A., Email: nknuth03@gmail.com
Prof. Dr. Andreas Nastansky, Hochschule für Wirtschaft und Recht (HWR) Berlin, Email: andreas.nastansky@hwr-berlin.de

Herausgeber: Prof. Dr. Andreas Nastansky, Hochschule für Wirtschaft und Recht (HWR) Berlin, Professur für Quantitative Methoden und Mathematik, Email: andreas.nastansky@hwr-berlin.de
2025, ISSN 0949-068X

Gründungs-herausgeber: Prof. Dr. Hans Gerhard Strohe, ehemals Lehrstuhl für Statistik und Ökonometrie, Wirtschafts- und Sozialwissenschaftliche Fakultät der Universität Potsdam

DOI: https://doi.org/10.25932/publishup-67486

Kurzfassung

Die Fähigkeit Volatilität präzise vorherzusagen, ist von zentraler Bedeutung für das Risikomanagement in Banken und für das Treffen fundierter Anlageentscheidungen. In diesem Beitrag wird der Einsatz von Deep-Learning-Methoden − speziell des Long Short-Term Memory (LSTM)-Netzwerkes − zur Prognose der Volatilität des Deutschen Aktienindex analysiert. Hierbei wird die Prognosegüte des LSTMs mit der von gängigen zeitreihenökonometrischen Ansätzen, wie den Generalized Autoregressive Conditional Heteroscedaticity (GARCH)-Modellen, verglichen. Obwohl LSTMs in vielen Bereichen zunehmend Anwendung finden, ist ihr Einsatz in der Vorhersage von Finanzmarktzeitreihen im Vergleich zu etablierten ökonometrischen Modellen noch wenig untersucht. Die empirischen Ergebnisse zeigen, dass das LSTM verschiedene symmetrische und asymmetrische GARCH-Modelle in Bezug auf die Vorhersagegenauigkeit sowohl im Trainings- als auch im Testdatensatz deutlich übertrifft. Künstliche Neuronale Netze bieten eine bessere Generalisierungsfähigkeit und niedrigere Prognosefehler. Gleichzeitig werden Herausforderungen des Einsatzes neuronaler Netze im stark regulierten Bankensektor diskutiert.

JEL-Klassifikation: C45, C58, G17

Schlagworte: Asymmetrische Volatilität, GARCH, LSTM, Künstliche Neuronale Netze, Volatilitätsprognosen

Abstract

In financial markets, the ability to accurately predict volatility is crucial for managing risk in banks and making informed investment decisions. This paper analyzes the use of deep learning methods − specifically the Long Short-Term Memory (LSTM) network − to forecast the volatility of the German stock index DAX. In this context, the predictive accuracy of the LSTM is compared with that of common time series econometric. Although LSTMs are increasingly used in many areas, their use in the prediction of financial time series is still little studied compared to established econometric models. The empirical results show that the LSTM network significantly outperforms various symmetric and asymmetric GARCH models in terms of in-sample and out-of-sample forecasting accuracy. Artificial neural networks offer better generalization capability and lower prediction errors. At the same time, challenges of using neural networks in the highly regulated banking sector are discussed.

JEL-Classification: C45, C58, G17

Keywords: Asymmetric Volatility, Artificial neural networks, GARCH, LSTM, Volatility Prediction

Inhaltsverzeichnis

Abkürzungsverzeichnis

Adam ..... Adaptive Moment Estimation
ADF ..... Augmented Dickey-Fuller
AIC ..... Akaike-Informationskriterium
ARCH ..... Autoregressiv Conditional Heteroskedasticity
BHHH ..... Berndt-Hall-Hall-Hausman
BIC ..... Bayes-Informationskriterium
BPTT ..... Backpropagation Trough Time
BSM ..... Black-Scholes-Merton-Modell, Black-Scholes-Merton-Modell
CBOE ..... Chicago Board Options Exchange
DAX ..... Deutscher Aktienindex
DDKZ ..... Demeterfi, Derman, Kamal und Zou
EGARCH ..... Exponential-GARCH
FNN ..... Feedforward Neural Network
GARCH ..... Generalized Autoregressive Conditional Heteroscedaticity
GJR-GARCH ..... Glosten-Jagannathan-Runkle-GARCH
GPU ..... Graphics Processing Unit
KNN ..... Künstliches Neuronales Netz
LSTM ..... Long Short-Term Memory
MAE ..... Mean Absolute Error
MaRisk ..... Mindestanforderungen an das Risikomanagement
MLE ..... Maximum Likelihood Estimation
MSE ..... Mean Squared Error
NAGARCH ..... Nonlinear-Asymmetric-GARCH
QML ..... Quasi-Maximum-Likelihood
QQ ..... Quantil-Quantil
ReLU ..... Rectified Linear Unit
RMSE ..... Root Mean Squared Error
RNN ..... Recurrent Neural Network
S&P ..... Standard and Poor's
SGD ..... Stochastic Gradient Descent
SLSQP ..... Sequential-Least-Squares-Programming
Tanh ..... Tangens hyperbolicus
TGARCH ..... Treshhold-GARCH
VDAX-NEW ..... DAX-Volatilitätsindex
VRP ..... Volatility-Risk-Premium

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Mit der Einführung der modernen Portfoliotheorie durch Harry S. Markowitz im Jahr 1952 können Anlageentscheidungen auf Basis einer modellierten Rendite-Risiko-Betrachtung getroffen werden.[1] Speziell institutionelle Finanzmarktakteure müssen kontinuierlich ihre Entscheidungen und Strategien auf der Grundlage variierender Einschätzungen zu Rendite und Risiko überdenken. Traditionell wird dabei das Risiko durch die beobachtete Volatilität veranschaulicht. Dahinter verbirgt sich eine Maßzahl für die Preisschwankung eines Gutes bzw. Basiswerts während einer bestimmten Zeitperiode. Die Volatilität kann im Laufe der Zeit erheblich variieren, wobei ruhige Phasen mit geringen Schwankungen von turbulenten Perioden mit starken Fluktuationen abgelöst werden. Trotz der frühen empirischen Erkenntnis sich ändernder Volatilitäten am Kapitalmarkt wurden noch lange Methoden genutzt, die von einer konstanten Volatilität ausgingen.

Dies änderte sich grundlegend mit einem Beitrag von Robert F. Engle aus dem Jahr 1982 zur Analyse der zeitlichen Entwicklung der britischen Inflationsrate.[2] Er führte das erste Modell von Residuen mit autoregressiv bedingter Heteroskedastizität ein, das auch die Eigenschaften vieler Finanzmarktzeitreihen gut erfassen konnte, indem die zeitlich variierende Volatilität adäquat modelliert werden konnte. Seine ARCH-Modelle und die daraus abgeleiteten verallgemeinerten ARCH (GARCH)-Modelle sind zu einem unverzichtbaren Werkzeugen in der Finanzmarktanalyse und im Risikomanagement von Finanzdienstleistern wie Banken geworden.

Parallel zur revolutionären Weiterentwicklung der Zeitreihenökonometrie durch Engle, standen neuronale Netzwerke noch vor erheblichen Herausforderungen. Dies führte zunächst zu einem Stillstand in der Anwendung und wissenschaftlichen Weiterentwicklung neuronaler Netze. Die Einführung des Backpropagation-Algorithmus im Jahr 1986 entfachte jedoch eine zweite Phase in der Entwicklung von leistungsfähigen Deep-Learning-Verfahren.[3] Ein entscheidender Durchbruch gelang 1997 mit der Entwicklung des Long Short-Term Memory-Netzwerks (LSTM) durch Sepp Hochreiter und Jürgen Schmidhuber. Allerdings konnte das Netzwerk sein volles Potenzial zunächst nicht entfalten, da noch nicht ausreichend Daten und Hardwareleistung vorhanden waren.

---

[1] Vgl. Markowitz (1952), S. 78f.
[2] Vgl. Engle (1982), S. 992.
[3] Vgl. Guo et al. (2020), S. 1460f.

Mit dem Aufkommen leistungsfähiger Hardware, insbesondere Grafikprozessoren (GPUs) und einer größeren Verfügbarkeit an Daten haben sich Deep-Learning-Verfahren zunehmend in der Finanzwirtschaft etabliert.[4] GPUs ermöglichen durch parallele Berechnungen eine schnelle Verarbeitung großer Datenmengen, was die Anwendung von LSTM-Netzwerken realisierbar macht. LSTMs sind besonders vielversprechend zur Modellierung und Prognose von Zeitreihen, da sie in der Lage sind, komplexe, nicht-lineare Muster zu erkennen und zu lernen.[5]

Der vorliegende Beitrag vergleicht die Eignung von LSTM-Netzwerken zur Volatilitätsprognose alternativ zu symmetrischen und asymmetrischen GARCH-Modellen. In der Theorie sollte das Deep-Learning-Verfahren aufgrund seiner nicht-parametrischen Eigenschaft dem zeitreihenökonometrischen Ansatz überlegen sein. GARCH-Modelle unterliegen zahlreichen Annahmen, die mehr oder weniger der Realität entsprechen. Durch den Verzicht auf diese Annahmen könnten LSTM-Netzwerke eine höhere Prognosegüte aufweisen, sofern es gelingt, die nicht-linearen Muster in den Daten korrekt zu erlernen.

Im ersten Teil wird das finanztheoretische Fundament gelegt, indem zunächst die Grundlagen der Volatilitätsmessung erörtert werden. Darauf aufbauend werden einige stilisierten Fakten zur historischen Volatilität in Abgrenzung zur impliziten Volatilität sowie die Indexberechnung des Volatilitätsindex VDAX-NEW beschrieben. Im Weiteren werden die genutzten zeitreihenökonometrischen Verfahren mit bedingter Heteroskedastizität vorgestellt. Zur Vertiefung wird auf die Maximum-Likelihood-Schätzung von GARCH-Modellen näher eingegangen, um die Bestimmung der ökonometrischen Parameter zu veranschaulichen. Der dritte Teil führt in die Anwendung von Deep Learning zur Volatilitätsprognose ein. Hierzu werden zunächst Neuronale Netze allgemein dargestellt und die Konzepte des Gradient Descent sowie der Backpropagation skizziert. Anschließend erfolgt die Überleitung zu den Recurrent Neural Networks (RNN) − bevor die LSTM-Netzwerke im Speziellen betrachtet werden. Dem folgt eine vergleichende Analyse der Schätzergebnisse und Prognosegüte am Beispiel des Volatilitätsindex VDAX-NEW für den Zeitraum Januar 1988 bis Mai 2024. Abschließend wird der Fokus auf die praktische Relevanz verbesserter Volatilitätsprognosen gelegt und es werden Herausforderungen beim Einsatz von Deep Learning im Bankensektor diskutiert.

[4] Vgl. Brown (2024), Absatz 2.
[5] Vgl. Hodjat (2015), Absatz 5.

2 Finanztheoretische Grundlagen

2.1 Volatilitätsmessung

Ausgangspunkt der Volatilitätsmessung bildet die Renditeberechnung. Eine gängige Methode zur Ermittlung der Rendite R_t eines Finanzinstruments zwischen zwei Zeitpunkten $t-1$ und t stellt die diskrete Rendite dar. Diese prozentuale Veränderung wird wie folgt berechnet:[6]

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}} = \frac{P_t}{P_{t-1}} - 1 \tag{2.1}$$

mit P_t als Preis des Finanzinstruments am Ende des Zeitraums *t* und P_{t-1} als Preis am Ende des vorangegangenen Zeitraums *t* – 1. Alternativ kann die logarithmische (stetige) Rendite verwendet werden, die die diskrete Rendite in eine additive Form umwandelt:[7]

$$r_t = \ln\left(\frac{P_t}{P_{t-1}}\right) = \ln(P_t) \; - \; \ln(P_{t-1}) \tag{2.2}$$

Aufgrund der Additivitätseigenschaft können die einperiodigen stetigen Renditen r_t durch Summation in mehrperiodische Renditen umgewandelt werden.[8] Da bei kleinen relativen Kursänderungen (wie bei der Nutzung von Tagesdaten von Finanzmarktpreisen üblich) diskrete und logarithmische Renditen praktisch übereinstimmen, wird für die empirische Analyse die logarithmische Variante herangezogen.

Aufbauend auf der Rendite lässt sich die Volatilität berechnen, welche sich grundsätzlich in historische und implizite (bzw. implizierte) Volatilität unterscheiden lässt. Die historische Volatilität misst die Schwankungsintensität des Preises eines Finanzinstruments über einen bestimmten Zeitraum. Sie ist ein gängiges Maß zur Einschätzung des Risikos und wird als Standardabweichung der logarithmischen Renditen berechnet. Die Standardabweichung σ für einen Zeitraum von N Tagen kann folgendermaßen ermittelt werden:[9]

$$\sigma = \sqrt{\sigma^2} = \sqrt{\frac{1}{N-1}\sum_{t=1}^{N}(r_t - \mu)^2} \tag{2.3}$$

Dabei ist r_t die logarithmische Rendite am Tag t und μ der Mittelwert der logarithmischen Renditen über den betrachteten Zeitraum.

---

[6] In Anlehnung an Bruns / Meyer-Bullerdiek (2020), S. 3f.
[7] Vgl. Hull (2015), S. 348.
[8] Vgl. Schmid / Trede (2006), S. 6.
[9] Vgl. Bruns / Meyer-Bullerdiek (2020), S. 10f.

Das Vorgehen zur Schätzung der historischen Volatilität in (2.3) wird als Close-to-Close-Volatilität bezeichnet und verwendet ausschließlich die täglichen Schlusskurse. Dabei werden Intraday-Informationen vernachlässigt.[10] Um detailliertere Schätzungen der Volatilität zu erhalten, wurden alternative Methoden wie die High-Low-Methode und die High-Low-Open-Close-Methode entwickelt. Die High-Low-Methode, eingeführt von Parkinson (1980), nutzt die Tageshöchst- und Tagestiefstpreise (H_t und L_t). Dabei wird ein Random Walk (Irrfahrtsprozess) für die Aktienpreisbewegung unterstellt. Die Diffusionskonstante, die den Random Walk charakterisiert, wird folglich zu der zentral zu bestimmenden Größe. Sie beschreibt die Varianz der Preisänderungen pro Zeiteinheit und charakterisiert die zufällige Bewegung eines Preises. Wird die historische Volatilität auf Schlusskursbasis bestimmt, so gleicht die Berechnung der Diffusionskonstante der einfachen Varianzformel. Liegen hingegen nur die Abstände zwischen tiefstem und höchstem Kurs vor, so müssen die Abstände mit dem Faktor $\frac{1}{4\ln(2)}$ multipliziert werden, um sicherzustellen, dass diese einem Random Walk entstammen. Der Abstand zwischen den Extremkursen entspricht ferner $\ln\left(\frac{H_t}{L_t}\right)$. Dadurch berücksichtigt diese Methode die Intraday-Preisschwankungen detaillierter als der Close-to-Close-Schätzer:[11]

$$\sigma_{HL} = \sqrt{\frac{1}{4\ln(2)N}\sum_{t=1}^{N}\ln\left(\frac{H_t}{L_t}\right)^2} \tag{2.4}$$

Letztlich stellt die High-Low-Open-Close-Methode von Garman und Klass (1980) eine Erweiterung der High-Low-Methode dar, indem sie auch Eröffnungs- und Schlusskurse (O_t und C_t) einbezieht:[12]

$$\sigma_{HLOC} = \sqrt{\frac{1}{N}\sum_{t=1}^{N}\left[\frac{1}{2}\ln\left(\frac{H_t}{L_t}\right)^2 - (2\ln 2 - 1)\ln\left(\frac{C_t}{O_t}\right)^2\right]} \tag{2.5}$$

Hierbei nutzt die Methode die Eigenschaften der geometrischen Brownschen Bewegung, um sowohl die Schwankungen innerhalb eines Tages als auch die Änderungen über Nacht zu berücksichtigen. Der Term $2\ln 2 - 1$ und $\frac{1}{2}$ sind dabei Gewichtungsfaktoren, um die beiden Komponenten bezüglich ihrer Bedeutung zu balancieren.[13]

[10] Vgl. Pesaran (2015), S. 412.
[11] Vgl. Parkinson (1980), S. 62f.
[12] Vgl. Garman / Klass (1980), S. 67ff.
[13] Vgl. ebenda.

Die praktische Bedeutung des High-Low-Open-Close-Schätzers besteht darin, dass siebenmal weniger Beobachtungen erforderlich sind, um die gleiche statistische Genauigkeit wie beim Close-to-Close-Schätzer zu erreichen. Wenngleich die gezeigten Erweiterungen die historischen Volatilitätsberechnung detaillierter darstellen, so muss darauf hingewiesen werden, dass die Erweiterungen strikten Annahmen folgen.[14] Diese Annahmen führen tendenziell zu geringeren Volatilitäten (siehe Abb. 1). Im Weiteren wird somit die Close-to-Close-Methode zur Volatilitätsberechnung herangezogen, um die Festlegung auf Annahmen der detaillierteren Schätzer zu vermeiden.

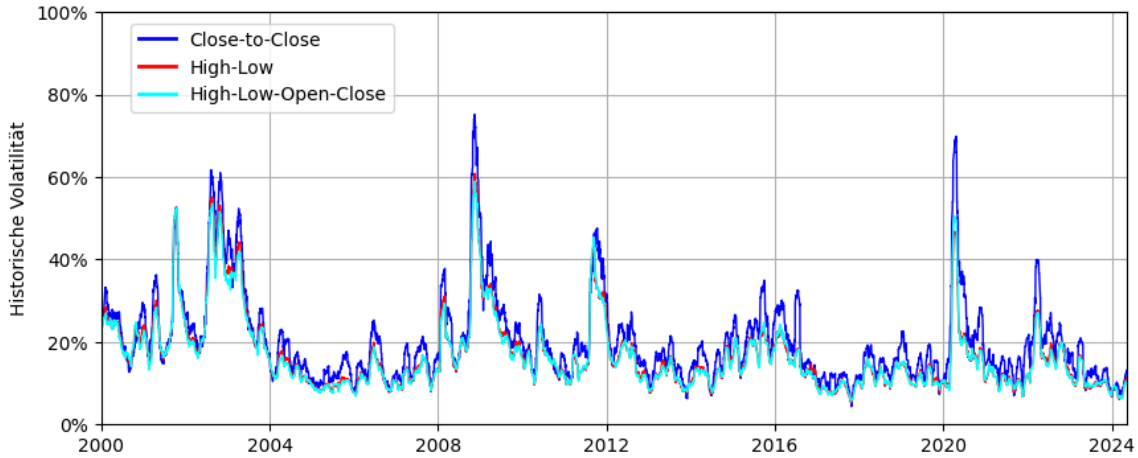

Abb. 1: Vergleich verschiedener Volatilitätsmessungen für den DAX
Quelle: Eigene Darstellung mit Daten der Onvista Media

Um diese Entscheidung zusätzlich zu begründen, wurden zwei statistische Tests durchgeführt. Um die Verzerrung der Schätzer zu überprüfen, wurde ein Einstichproben-t-Test verwendet. Der t-Test prüft die Nullhypothese, dass der Mittelwert der Differenzen zwischen den geschätzten Volatilitäten (High-Low und High-Low-Open-Close) und der Referenzvolatilität (Close-to-Close) gleich Null ist. Ein p-Wert von nahe 0 führt zur Testentscheidung, dass diese Nullhypothese verworfen wird, was auf eine signifikante Verzerrung hindeutet. Die Ergebnisse des t-Tests zeigen, dass sowohl der High-Low-Schätzer (t-Statistik $= -84{,}51$; p-Wert ≈ 0) als auch der High-Low-Open-Close-Schätzer (t-Statistik $= -85{,}45$; p-Wert ≈ 0) signifikant von der Referenzvolatilität abweichen.[15]

[14] Vgl. Li / Weinbaum (2000), S. 6f.
[15] Vgl. Hartung (2009), S. 179.

Zusätzlich wurde ein Levene-Test auf Varianzhomogenität durchgeführt, um zu prüfen, ob die Varianzen der Volatilitätsschätzungen gleich sind. Unterschiede in der Varianz können auf unterschiedliche Effizienzen der Schätzer hinweisen. Der Levene-Test prüft die Nullhypothese, dass die Varianzen der Volatilitätsschätzungen gleich sind. Ein p-Wert von ungefähr 0 führt zur Testentscheidung, dass die Nullhypothese verworfen wird. Das weist darauf hin, dass die Varianz der Schätzer unterschiedlich sind und somit unterschiedliche Effizienzen aufweisen. Die Ergebnisse des Levene-Tests zeigen, dass der High-Low-Schätzer (F-Statistik = 65,58; p-Wert ≈ 0) und der High-Low-Open-Close-Schätzer (F-Statistik = 84,50; p-Wert ≈ 0) jeweils im Vergleich zum Close-to-Close-Schätzer unterschiedliche Varianzen aufweisen.[16]

Die implizite Volatilität ist hingegen ein Maß für die erwartete zukünftige Volatilität eines Finanzinstruments, das aus den Preisen von gehandelten Optionen abgeleitet wird.[17] Sie reflektiert damit die Markterwartungen über die zukünftige Preisbewegung des zugrundeliegenden Finanzinstruments. Das Black-Scholes-Merton-Modell (BSM) bietet eine Möglichkeit zur Berechnung der impliziten Volatilität. Unter Annahmen wie konstanten Zinssätzen, einem log-normal verteilten Preis des Underlyings und der Abwesenheit von Arbitragemöglichkeiten lautet die Preisformel für eine europäische Call-Option:[18]

$$\begin{aligned} C &= S_0 N(d_1) - K e^{-r_f T} N(d_2) \\ d_1 &= \frac{\ln\left(\frac{S_0}{K}\right) + \frac{r_f + \sigma^2}{2} T}{\sigma\sqrt{T}} \\ d_2 &= d_1 - \sigma\sqrt{T} \end{aligned} \tag{2.6}$$

Hierbei steht C für den Preis der Call-Option, S_0 für den aktuellen Preis des Underlyings, K für den Basispreis, r_f für den risikolosen Zinssatz, T für die Restlaufzeit der Option und $N(.)$ für die kumulative Verteilungsfunktion der Standardnormalverteilung.[19]

Da das BSM-Gleichungssystem nicht direkt nach σ aufgelöst werden kann, wird die implizite Volatilität iterativ durch Verfahren wie das Newton-Raphson-Verfahren bestimmt. Dabei handelt es sich um ein numerisches Verfahren, bei dem eine Näherungslösung schrittweise verbessert wird.[20]

[16] Vgl. Backhaus et al. (2021), S. 217f.
[17] Vgl. Mauer / Nastansky (2024), S. 7.
[18] Vgl. Bloss (2020), S. 209f; Hull (2015), S. 343ff.
[19] Vgl. Hull (2015), S. 357ff.
[20] Vgl. Bloss (2020), S. 200f.

Die Differenz zwischen historischer und impliziter Volatilität – auch als Volatility-Risk-Premium (VRP) bezeichnet – repräsentiert dabei eine Prämie, die die Risikobereitschaft der Marktteilnehmer widerspiegelt und als eine Art Versicherung gegenüber Marktturbulenzen interpretiert werden kann.[21] Eine positive Differenz bedeutet, dass die implizite Volatilität höher ist als die tatsächliche historische Volatilität. Dies deutet darauf hin, dass Marktteilnehmer bereit sind, eine Prämie zu zahlen, um sich gegen unvorhergesehene Marktbewegungen abzusichern.

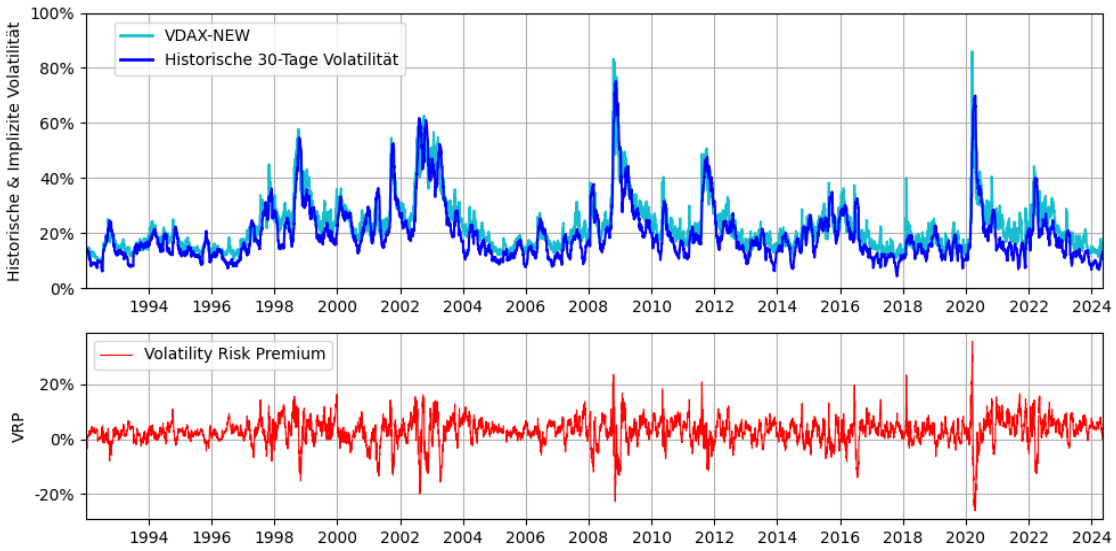

Abb. 2: Volatility-Risk-Premium
Quelle: Eigene Darstellung mit Daten der Onvista Media

2.2 Stilisierte Fakten von Renditen und Volatilität

Die Renditen von liquide handelbaren Finanzinstrumenten haben typische Eigenschaften, die als stilisierte Fakten bezeichnet werden. Zu diesen zählen u.a. Asymmetrien, hohe Autokorreliertheit der absoluten Renditen, kontemporäre Korrelationen, schwere Flanken und Spitzigkeit sowie Volatilitätscluster.[22] Üblicherweise ist die empirische Verteilung der Renditen von Finanzanlagen nicht völlig symmetrisch, sondern zeigt eine schwache Asymmetrie. Stark negative Renditen treten etwas häufiger auf als stark Positive. Ein weiteres Merkmal der Renditen von Finanzmarktzeitreihen ist das Auftreten von schweren Verteilungsenden (Fat Tails), d. h., dass die Verteilungsenden der Renditen mehr Masse aufweisen als bspw. eine Normalverteilung. Konkret bedeutet das, dass extreme Preisbewegungen häufiger auftreten, als es bei einer normalverteilten Zufallsvariable der Fall wäre. Hinzu kommt, dass betragsmäßig sehr kleine

[21] Vgl. Ang et al. (2018), S. 3.
[22] Vgl. Schmid / Trede (2006), S. 12f.; Barz / Nastansky (2024), S. 15.

Renditen häufiger nachzuweisen sind (Spitzigkeit). Die Verteilung der beobachtbaren Renditen ist dann leptokurtisch. Die Kurtosis repräsentiert eine statistische Kennzahl für das kombinierte Gewicht der Enden einer Verteilung im Verhältnis zum Zentrum der Verteilung. Abb. 3 veranschaulicht diesen Effekt, da die Verteilung der empirischen logarithmischen Renditen stärker mit der t-Verteilung als mit einer Normalverteilung (mesokurtische Verteilung) übereinstimmt. Dies deutet darauf hin, dass die tatsächlichen Renditen eine höhere (positive) Kurtosis und somit ausgeprägtere Extremwerte innehaben und somit das Finanzinstrument anfällig für extreme Preisbewegungen ist. Diese Beobachtung wird ferner in den QQ-Plots durch die starken Abweichungen von der Referenzlinie deutlich.

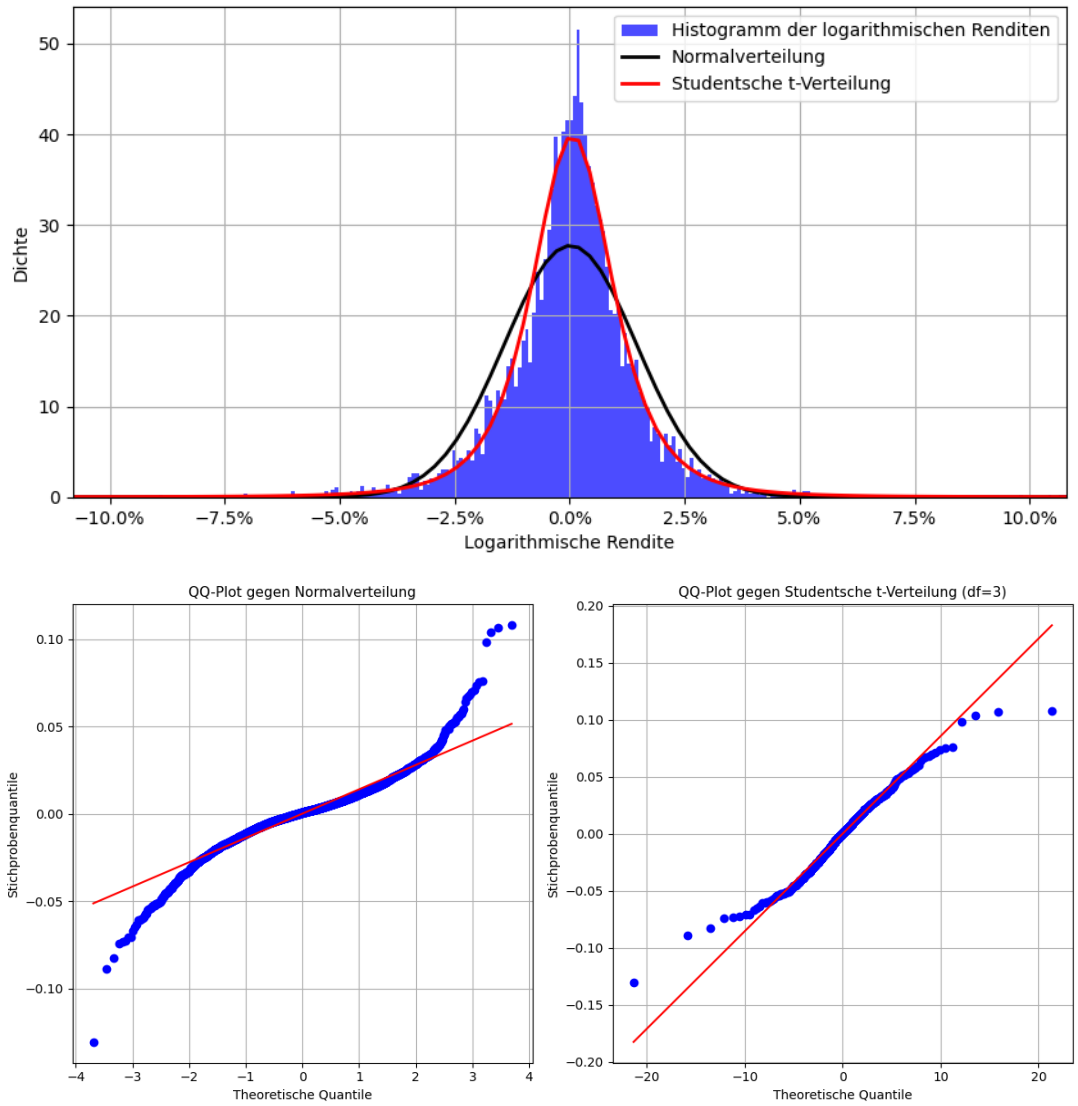

Abb. 3: Verteilung und QQ-Plot der täglichen logarithmischen DAX-Renditen
Quelle: Eigene Darstellung mit Daten der Onvista Media

Eine weitere Eigenschaft von Finanzzeitreihen ist, dass die Abhängigkeiten zwischen den Renditen insbesondere bei Tagesdaten zeitlich sehr schnell verschwinden. Die Renditen

sind schon nach einer sehr kurzen Zeitspanne vollständig unkorreliert.[23] Abb. 4 zeigt in der Autokorrelationsfunktion, dass die logarithmischen Renditen nahezu keine Autokorrelation aufweisen, während sich die absoluten Renditen durch eine signifikante und allmählich abnehmende Autokorrelation über mehrere Zeitperioden (Lags) hinweg auszeichnen.

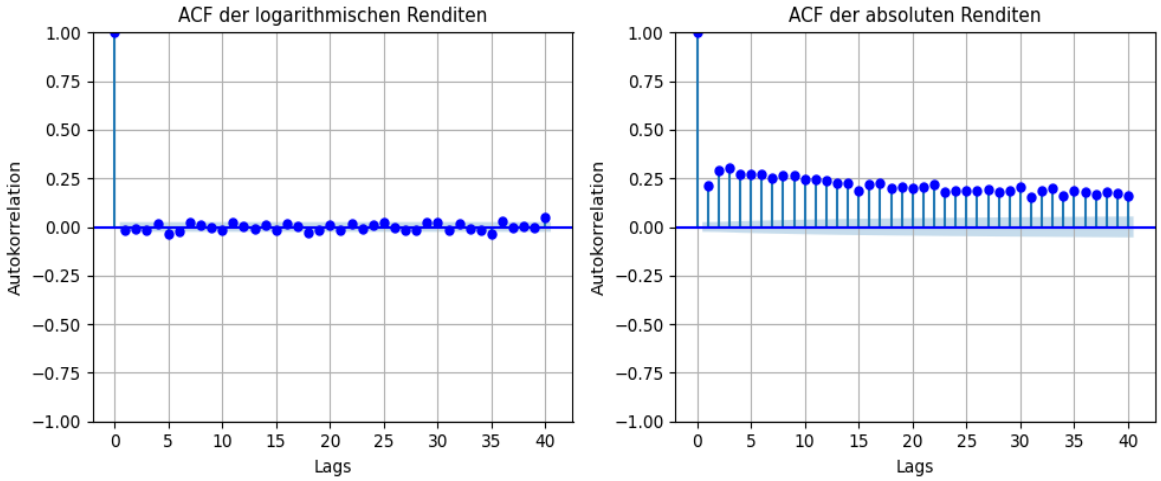

Abb. 4: Autokorrelationsfunktion der absoluten und logarithmischen DAX-Renditen
Quelle: Eigene Darstellung mit Daten der Onvista Media

Das Phänomen, dass große absolute $|r_t|$ oder quadratische Renditen r_t^2 tendenziell aufeinanderfolgen, damit hoch korreliert sind und langsam Abklingen, ist eng mit dem stilisierten Fakt der Volatilitätscluster (Volatility Clustering) verbunden.[24] Dabei wechseln sich Phasen hoher und geringer Volatilität unregelmäßig ab – auch als bedingte Heteroskedastizität bekannt. Auf einen Tag mit hohen Ausschlägen folgen dann weitere Tage hoher Kursschwankungen. Dasselbe gilt umgekehrt. Daraus folgt, dass die Volatilität nicht gleichmäßig über die Zeit verteilt ist, sondern in Clustern auftritt. Dieses Phänomen wird durch Abb. 5 illustriert, da die historische Volatilität über den dargestellten Zeitraum nicht konstant bleibt, sondern deutliche Wechsel zwischen hohen und niedrigen Niveaus aufweist. Aus diesem empirischen Befund kann die Vermutung abgeleitet werden, dass sich die Volatilität der Finanzinstrumente besser aus ihrer Vergangenheit prognostizieren lässt als ihre Rendite- bzw. Kursentwicklung.[25] Für die Bewertung von Optionen spielt diese Beobachtung eine wichtige Rolle.

[23] Vgl. Schmid / Trede (2006), S. 14.
[24] Vgl. Brooks (2008), S. 380.
[25] Vgl. Mauer / Nastansky (2024), S. 25ff.

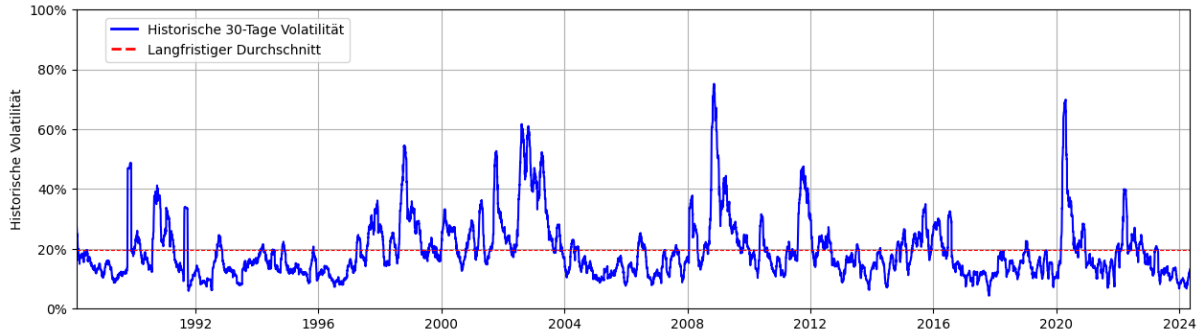

Abb. 5: Historische 30-Tage Volatilität des DAX
Quelle: Eigene Darstellung mit Daten der Onvista Media

Überdies wird aus Abb. 5 deutlich, dass die Volatilität dazu neigt, zu einem langfristigen Mittelwert zurückzukehren. Hohe Volatilitätsniveaus werden tendenziell abnehmen, während niedrige Volatilitätsniveaus zunehmen − bis sie das langfristige Mittel erreichen. Diese Eigenschaft ist auch als Mean Reverting Behaviour of Volatility bekannt. In Abb. 6 wird veranschaulicht, wie die Volatilität über verschiedene Zeiträume nach einer Beobachtung in den oberen und unteren 10 Prozent des historischen Verteilungsbereichs konvergiert. Interessanterweise kehren die oberen 10 Prozent deutlich schneller zum langfristigen Durchschnitt zurück als die unteren 10 Prozent. Dies deutet darauf hin, dass niedrigere Volatilitätsniveaus deutlich stabiler und beständiger sind als hohe Volatilitätsniveaus. Gleichermaßen zeigt sich diese Beobachtung in der rechtsschiefen Verteilung der Volatilität, d. h., dass es mehr Perioden hoher Volatilität gibt, als es bei einer normalverteilten Zufallsvariable der Fall wäre.[26] Ein empirisches Volatilitätsmodell sollte daher in der Lage sein, die skizzierten stilisierten Fakten adäquat zu erfassen und wiederzugeben.

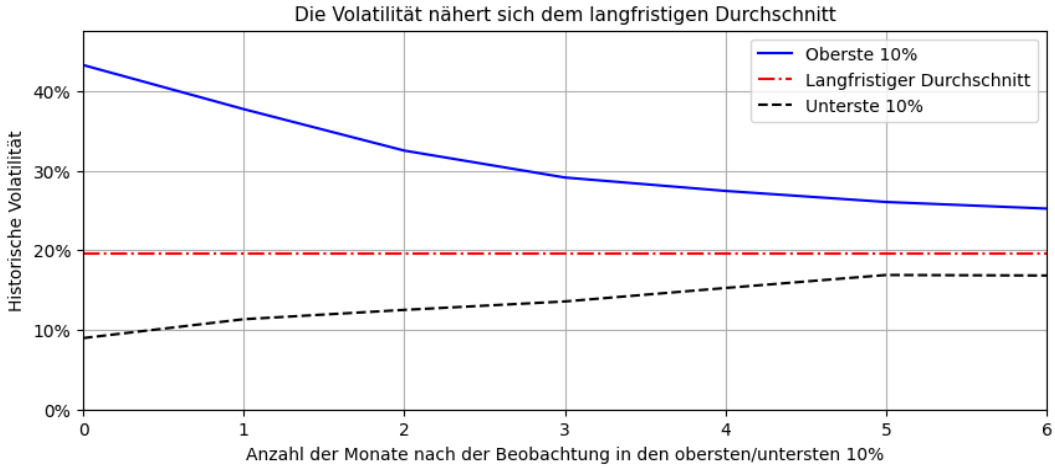

Abb. 6: Mean Reverting Behavior of Volatility
Quelle: Eigene Darstellung mit Daten der Onvista Media

[26] Vgl. Poon / Granger (2003), S. 482f.

2.3 Indexberechnung des VDAX-NEW

Der Volatilitätsindex VDAX-NEW wurde 2005 von der Deutschen Börse eingeführt und ersetzte damit den bisherigen VDAX. Beide Indizes sind als aus Optionspreisen abgeleitete (implizite) Volatilität definiert. Die Einführung orientierte sich stark am Konzept des VIX, dem Volatilitätsindex der Chicago Board Options Exchange (CBOE), der bereits 2003 neu konzipiert wurde. Ziel des VDAX-NEW ist es, die vom Markt erwartete Volatilität des DAX über einen Zeitraum von 30 Tagen abzubilden. Dabei fließt eine Vielzahl von at-the-money und out-of-the-money DAX-Optionen unterschiedlicher Preisklassen und Fälligkeiten in die Berechnung ein, um ein umfassendes Bild der Marktvolatilität zu erzeugen.[27]

Die finanztheoretische Grundlage der Indexberechnung bildet die von Demeterfi et al. (DDKZ) entwickelte Theorie des Fair Value of Future Variance. Im Gegensatz zur impliziten Volatilität gemäß des Black-Scholes-Merton-Modell wird die DDKZ-Varianz direkt aus den Optionspreisen wie folgt extrahiert:[28]

$$\sigma_{ddkz}^2 = \frac{2}{T}\left(r_f T - \left[\frac{S_0}{S_*}e^{r_f T} - 1\right] - \ln\left(\frac{S_*}{S_0}\right) + e^{r_f T}\int_0^{S_*}\frac{P(T,K)}{K^2}dK + e^{r_f T}\int_{S_*}^{\infty}\frac{C(T,K)}{K^2}dK\right) \tag{2.7}$$

In der Kerngleichung dieser Theorie repräsentiert S_0 den aktuellen Preis des Basiswerts; während $C(T,K)$ und $P(T,K)$ die Preise von Call- und Put-Optionen mit Laufzeit T und Basispreis K darstellen. Der beliebig wählbare Preis S_* legt die Grenze zwischen den betrachteten Call- und Put-Optionen fest. Hinter (2.7) steht die Idee der Konstruktion eines Portfolios von Optionen zur Replikation der Varianz. Dazu wird eine Short-Position in $\frac{1}{S_*}$ Forward-Kontrakten mit Basispreis S_* und ein Kontinuum von Long-Positionen in $\frac{1}{K^2}$ Put-Optionen für alle Basispreise von 0 bis S_* sowie $\frac{1}{K^2}$ Call-Optionen mit Basispreis S_* bis ∞ kombiniert. Durch die invers proportionale Gewichtung mit $\frac{1}{K^2}$ ergibt sich ein Portfolio, welches vom Preis des Underlyings unabhängig auf Varianzänderungen reagiert. Damit stellt σ_{ddkz}^2 den fairen Swap-Satz eines Varianz-Swaps dar, bei dem am Ende der Laufzeit T die historische Varianz gegen den vereinbarten Swap-Satz getauscht wird.[29]

[27] Vgl. Deutsche Börse (2018), S. 5f.
[28] Vgl. Jiang / Tian (2007), S. 3.
[29] Vgl. Demeterfi et al. (1999), S. 19ff.

In der praktischen Anwendung wird (2.7) durch eine diskrete Summierung approximiert, um die Berechnung zu vereinfachen und auf die tatsächlich gehandelten Optionen anzuwenden, da nur endlich viele Basispreise K vorliegen:[30]

$$\sigma^2 = \frac{2}{T}\sum_j \frac{\Delta K_j}{K_j^2} e^{r_f T} M(K_j) - \frac{1}{T}\left(\frac{F}{K_0} - 1\right)^2$$
$$\Delta K_j = \frac{K_{j+1} - K_{j-1}}{2} \tag{2.8}$$

Hierbei steht F für den Forward-Preis des DAX, K_j für den Basispreis der j-ten Option (Call für $K_j > F$ bzw. Put für $K_j < F$). $M(K_j)$ gibt den Preis der Option mit Basispreis K_j an. $M(K_0)$ ist der Durchschnitt aus Call und Put am Basispreis K_0 (größter Basispreis unterhalb des Forward-Preis F). Der Forward-Preis wird hier über die Put-Call-Parity bestimmt. Die Gewichtung erfolgt identisch wie zu (2.7). Beim Vergleich von (2.7) und (2.8) zeigt sich, dass der erste Term in der praktischen Approximation die numerische Integration zur Berechnung der Integrale darstellt. Der zweite Term derselben Gleichung resultiert aus den vor den Integralen in (2.7) stehenden Termen, wobei die Logarithmusfunktion über eine Taylorreihenentwicklung approximiert wird.[31]

Für die Berechnung des VDAX-NEW werden mit Hilfe von (2.8) zunächst acht Subindizes einer DAX-Optionsserie mit Fälligkeiten in den kommenden zwei Jahren ermittelt. Der standardisierte VDAX-NEW wird im zweiten Schritt durch Interpolation der die 30 Tage umschließenden Subindizes σ_i^2 und σ_{i+1}^2 mit einer normierten Restlaufzeit von T_i und T_{i+1} berechnet. Falls dies nicht möglich ist, z. B. wenn es keine umschließenden Subindizes gibt, wird extrapoliert. Die Berechnung des VDAX-NEW als laufzeitunabhängiger Gesamtindex ergibt sich dementsprechend durch:[32]

$$\text{VDAX-NEW} = 100 \cdot \sqrt{\left[\bar{t}_i\, \sigma_i^2 \left(\frac{N_{Ti+1}-30}{N_{Ti+1}-N_i}\right) + \bar{t}_{i+1}\, \sigma_{i+1}^2 \left(\frac{30-N_{Ti}}{N_{iT+1}-N_{Ti}}\right)\right]\frac{365}{30}} \tag{2.9}$$

Die Interpolation der Subindizes σ_i^2 und σ^2_{i+1} mit der Restlaufzeit T_i bzw. T_{i+1}, wobei $N_{T\,i}$ die Zeit in Tagen bis zum Verfall der i-ten DAX-Optionsserie und $\bar{t}_i$ die normierte Restlaufzeit $\bar{t}_i = N_{Ti}/365$ angibt. Der VDAX-NEW repräsentiert somit eine Approximation der zukünftigen Volatilität.

[30] Vgl. Deutsche Börse (2007), S. 17.
[31] Vgl. ebenda.
[32] Vgl. Tallau (2012), S. 4.

3 Ökonometrische Verfahren der Volatilitätsmodellierung

3.1 ARCH-Modell

ARCH (Autoregressive Conditional Heteroskedasticity)-Modelle stellen ein bedeutendes Instrument zur Analyse der Volatilität von Finanzmarktzeitreihen dar. Das Akronym ARCH verdeutlicht, dass die bedingte Varianz eines ARCH-Prozesses von den vergangenen Beobachtungen des Prozesses selbst abhängt (autoregressives Modell - AR). Beim Auftreten von Volatilitätsclustern ist die Varianz der Renditen nicht konstant, sondern variiert über die Zeit (bedingte Heteroskedastizität - CH). Liegt beispielsweise eine Phase hoher Volatilität vor und nimmt die Volatilität am Tag *t* einen hohen Wert an, so kann eher davon ausgegangen werden, dass die Volatilität am Tag *t* + 1 ebenfalls hoch sein wird. Formal lässt sich das ARCH-Modell in allgemeiner Form für Finanzmarktrenditen darstellen durch:[33]

$$\begin{aligned} r_t &= \mu + \epsilon_t \\ \epsilon_t &= \sigma_t v_t \\ \sigma_t^2 &= \alpha_0 + \sum_{i=1}^{p} \alpha_i \epsilon_{t-i}^2 \end{aligned} \tag{3.1}$$

Dabei ist r_t die Rendite eines Finanzinstruments, μ der Mittelwert der Renditen und ϵ_t der unerwartete Kursausschlag des Finanzinstruments bzw. das Residuum zum Zeitpunkt t. Dieses Residuum ist über die verschiedenen Lags unkorreliert. Die erste Gleichung in (3.1) beschreibt den Erwartungswert der Rendite. Im einfachsten Fall wird die erwartete Rendite μ als konstant angenommen. Für den weiteren Verlauf dieser Arbeit ist diese erste Gleichung des ARCH-Modells zu vernachlässigen. Der stochastische Prozess v_t bildet einen unabhängig und identisch verteilter (i.i.d.) White-Noise-Prozess mit $E(v_t) = 0$ und $Var(v_t) = 1$ ab − für den meist die (Standard-)Normalverteilung angenommen wird. Die Varianz σ_t^2 ist konditional bedingt auf die Kenntnis der vergangenen Renditebeobachtungen: $Var(r_t|r_{t-1}, \ldots, r_{t-p})$. Die bedingte Varianz wird als eine lineare Funktion von p zeitverzögerten quadrierten Residuen modelliert. Weil σ_t^2 nicht negativ sein kann, gelten die Nichtnegativitätsbedingungen $\alpha_0 > 0$ und $\alpha_i > 0$ für $i = 1, \ldots, p$. Daraus folgt, dass hohe vergangene Residuen hohe bedingte Varianzen zum Zeitpunkt *t* nach sich ziehen und somit das Volatility Clustering nachbilden.[34]

[33] In Anlehnung an Schmitt (2012), S. 279.
[34] Vgl. Degiannakis / Xekalaki (2004), S. 10f.

Ein ARCH(p)-Prozess muss die (schwache) Stationaritätsbedingung erfüllen, da sonst die bedingte Varianz explodiert. Stationär ist der Prozess dann, wenn die Summe der autoregressiven Modellparameter ohne Konstante kleiner als eins ist ($\sum_{i=1}^{p} \alpha_i < 1$). In diesem Fall konvergiert die unbedingte Varianz. Durch wiederholtes Einsetzen und Anwenden des Gesetzes des iterierten Erwartungswertes folgt daher $\overline{\sigma}^2 = \frac{\alpha_0}{1-\sum_{i=1}^{p} \alpha_i}$ für die unbedingte Varianz. Ein einfaches Beispiel soll die Funktionsweise der ARCH(p)-Modelle demonstrieren. Für die Ordnung $p = 1$ ergibt sich die bedingte Varianz σ_t^2:[35]

$$\begin{aligned} \sigma_t^2 &= \alpha_0 + \alpha_1 \epsilon_{t-1}^2 = \frac{\alpha_0}{1-\alpha_1} + \alpha_1 \left(\epsilon_{t-1}^2 - \frac{\alpha_0}{1-\alpha_1} \right) \\ &= \overline{\sigma}^2 + \alpha_1 \left(\epsilon_{t-1}^2 - \overline{\sigma}^2 \right) \end{aligned} \tag{3.2}$$

Folglich ist die bedingte Varianz größer oder kleiner als die unbedingte Varianz, wenn das Quadrat der letzten Beobachtung größer respektive kleiner als die unbedingte Varianz ist. Somit schwankt die bedingte Varianz um die unbedingte Varianz. In Phasen hoher Marktvolatilität steigt die bedingte Varianz und kann demnach auch als Risikomaß interpretiert werden. In der Praxis kann die unbedingte Varianz als langfristiges Durchschnittsniveau interpretiert werden. Das ist auch als Mean Reversion bekannt. (3.2) zeigt ferner, dass das ARCH-Modell Volatilitätsschwankungen adäquat abbildet, da nach großen unerwarteten Kursänderungen eine hohe Volatilität erwartet wird und umgekehrt.[36]

Das letzte Merkmal zur Beschreibung der Form einer Verteilung ist die Kurtosis, die durch das vierte Moment der Verteilung definiert wird. Die Existenz dieses vierten Moments ist entscheidend für die Bestimmung der Kurtosis eines ARCH-Modells. Im ARCH(1)-Modell muss dabei die Bedingung $3a_1^2 < 1$ erfüllt sein, damit dieses Moment existiert. Für die (unbedingte) Kurtosis eines ARCH(1)-Modells gilt dann:[37]

$$\text{Kurt}(\epsilon_t) = \frac{E[\epsilon_t^4]}{E[\epsilon_t^2]^2} = 3\frac{1-\alpha_1^2}{1-3\alpha_1^2} > 3 \tag{3.3}$$

Ein Wert von drei entspricht der Kurtosis einer Normalverteilung und bei Vorhandensein des vierten Moments ist das Verhältnis stets größer als eins. Folglich ist die unbedingte Verteilung leptokurtisch. Dies lässt sich auch für ARCH-Prozesse höherer Ordnung zeigen. Somit bildet das ARCH-Modell auch die massereichen Verteilungsenden der Volatilität adäquat ab.

---

[35] Vgl. Schmitt (2012), S. 280.
[36] Vgl. Engle (1982), S. 992.
[37] Vgl. Schmitt (2012), S. 281.

3.2 GARCH-Modell

Wenngleich das ARCH-Modell die Volatilität theoretisch gut modelliert, wurde bei empirischen Anwendungen auf Finanzzeitreihen festgestellt, dass zur Erreichung einer adäquaten Anpassung eine hohe Ordnung p notwendig ist. Dementsprechend wären $p + 1$ Parameter zu schätzen. Ein häufiges Problem bei der Schätzung von ARCH-Modellen mit hohen Ordnungen besteht darin, die Nichtnegativitätsbedingungen der Parameter der Varianzgleichung einzuhalten. Zudem kollidiert diese Beobachtung mit dem von Box und Jenkins formulierte Sparsamkeitsprinzip, welches besagt, dass ein Modell so einfach wie möglich sein sollte, solange es eine angemessene Anpassung an die Daten bietet.[38] Daraus folgt, dass ein Volatilitätsmodell nur so viele Parameter enthalten sollte − wie unbedingt nötig, um die wesentlichen Eigenschaften der Daten zu erfassen. Um diesem Prinzip gerecht zu werden und zugleich eine flexiblere Lag-Struktur zu erlauben, entwickelte Tim Bollerslev 1986 das Generalized ARCH (GARCH)-Modell.[39]

Die Verallgemeinerung des ARCH-Modells sieht vor, dass die bedingte Varianz der Residuen neben den p verzögerten quadrierten Residuen auch von der bedingten Varianz der bis zu q-Vorperioden σ_{t-j}^2 abhängt. Somit wird die bedingte Varianz eines GARCH(p,q)-Modells wie folgt modelliert:[40]

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2 \tag{3.4}$$

Anders als beim ARCH(p)-Prozess wird nicht nur auf die letzten p Werte des Prozesses, sondern zusätzlich auf seine gesamte Vergangenheit bedingt. Die Nichtnegativitätsbedingungen $\alpha_0 > 0$, $\alpha_i > 0$ für $i = 1, \dots, p$ und $\beta_j > 0$ für $j = 1, \dots, q$ garantieren erneut, dass die bedingte Varianz strikt positiv wird.[41]

Gleichermaßen wie der ARCH(p)-Prozess setzt auch der GARCH(p,q)-Prozess das Vorliegen von (schwacher) Stationarität voraus. Gemäß dem ersten Theorem von Bollerslev ist ein GARCH-Prozess genau dann schwach stationär, wenn die Parameterrestriktion $\sum_{i=1}^{p} \alpha_i + \sum_{j=1}^{q} \beta_j < 1$ gilt.[42]

---

[38] Vgl. Box et al. (2016), S. 14f.
[39] Vgl. Schmid / Trede (2006), S. 175.
[40] Vgl. Schmitt (2012), S. 283.
[41] Vgl. Neusser / Wagner (2022), S. 168.
[42] Vgl. Bollerslev (1986), S. 310.

Die unbedingte Varianz $\overline{\sigma}^2$ eines stationären GARCH-Prozesses beträgt:[43]

$$\overline{\sigma}^2 = \frac{\alpha_0}{1 - \sum_{i=1}^{p} \alpha_i - \sum_{j=1}^{q} \beta_j} \tag{3.5}$$

Für die Ordnung $p = 1$ und $q = 1$ kann durch rekursives Ersetzen von σ_{t-1}^2 gezeigt werden, wie aus dem GARCH(1,1)-Modell ein äquivalentes ARCH(∞)-Modell mit geometrisch fallenden Gewichten in der Varianzfunktion wird:[44]

$$\begin{aligned} \sigma_t^2 &= a_0 + a_1 \epsilon_{t-1}^2 + \beta_1 \sigma_{t-1}^2 \\ &= \mathrm{a_0} + \mathrm{a_1}\epsilon_{\mathrm{t-1}}^2 + \beta_1(\mathrm{a_0} + \mathrm{a_1}\epsilon_{\mathrm{t-2}}^2 + \beta_1\sigma_{\mathrm{t-2}}^2) \\ &= \mathrm{a_0} + \mathrm{a_1}\epsilon_{\mathrm{t-1}}^2 + \beta_1\mathrm{a_0} + \beta_1\mathrm{a_1}\epsilon_{\mathrm{t-2}}^2 + \beta_1^2\sigma_{\mathrm{t-2}}^2 \\ &= \cdots = \frac{\mathrm{a_0}}{1-\beta_1} + \mathrm{a_1}\sum_{\mathrm{i=1}}^{\infty} \beta_1^{\mathrm{i-1}}\epsilon_{\mathrm{t-i}}^2 \end{aligned} \tag{3.6}$$

Dieser Zusammenhang kann auch für den allgemeinen GARCH(p,q)-Prozess formuliert werden:[45]

$$\sigma_t^2 = \frac{a_0}{1 - \sum_{i=1}^{p} \beta_i} + a_1 \sum_{i=1}^{\infty} \lambda_i \epsilon_{t-i}^2 \tag{3.7}$$

wobei die Parameter λ_i eine Funktion von α_i und β_j sind. Somit wird die Intension von Bollerslev deutlich, dass ein GARCH-Modell ein ARCH-Modell hoher Ordnung vereinfacht darstellt. Folglich gilt auch hier, dass das Volatility Clustering adäquat abgebildet werden kann.

Analog zum ARCH-Modell ist die Existenz des vierten Moments auch für die Bestimmung der Kurtosis bei einem GARCH-Modell entscheidend. Im GARCH(1,1)-Modell muss dafür die Bedingung $3\alpha_1^2 + 2\alpha_1\beta_1 + \beta_1^2 < 1$ erfüllt sein. Daraus ergibt sich die Kurtosis eines GARCH(1,1)-Modells wie folgt:[46]

$$\mathrm{Kurt}(\epsilon_t) = \frac{E[\epsilon_t^4]}{E[\epsilon_t^2]^2} = \frac{6\alpha_1^2}{1 - \beta_1^2 - 2\alpha_1\beta_1 - 3\alpha_1^2} \tag{3.8}$$

Im Falle eines stationären Prozesses ist die Kurtosis größer drei, woraus ein leptokurtisches Verhalten wie beim ARCH(1)-Modell folgt.[47]

---

[43] Vgl. Schmitt (2012), S. 284.
[44] Vgl. ebenda.
[45] Vgl. Bera / Higgins (1993), S. 313.
[46] Vgl. Bollerslev (1986), S. 312.
[47] Vgl. ebenda.

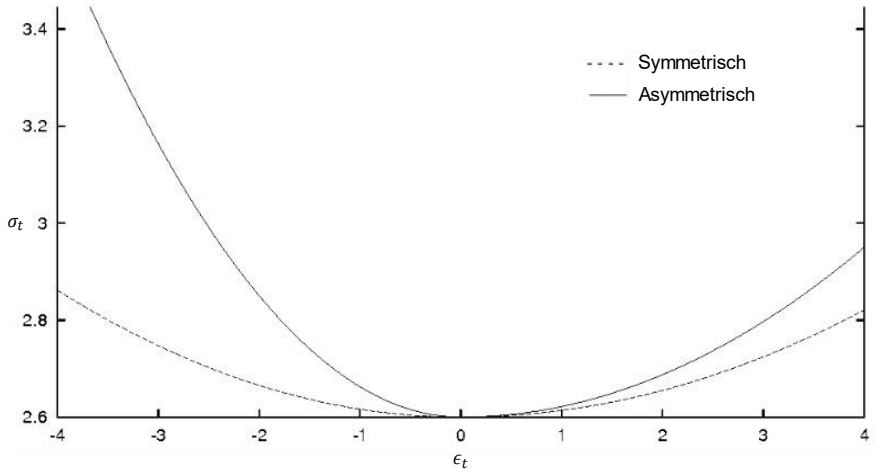

Abb. 7: Asymmetrie der Volatilität
Quelle: Eigene Darstellung in Anlehnung an Chen (2019), S. 9

Sowohl ARCH- als auch GARCH-Modelle basieren auf der Annahme, dass die bedingte Varianz symmetrisch auf positive und negative Kursausschläge ϵ_t reagiert. Das bedeutet, dass ein Kursanstieg denselben Einfluss auf die Volatilität hat wie ein Kursrückgang derselben Größenordnung. Dies steht im Widerspruch zum empirisch beobachtbaren Leverage-Effekt bei Aktienrenditen.[48] Dieser besagt, dass die Volatilität bei Kursrückgängen tendenziell stärker ansteigt als bei Kursanstiegen gleicher Größe (siehe Abb. 7). Eine Begründung dieser Beobachtung basiert auf der Annahme, dass eine Änderung der Aktienkurse sowohl die aktuelle als auch die zukünftig erwartete Volatilität beeinflusst. Ist der durch eine Veränderung der Aktienkurse verbundene Anstieg der Volatilität verfestigt und wirkt sich im Weiteren auch auf die erwartete Volatilität aus, so trägt eine negative Rendite zu einem weiteren Rückgang der Aktienkurse bei – aufgrund der gestiegenen Volatilität. Bei einer positiven Rendite kompensiert der beschriebene Effekt hingegen einen Teil der Kursanstiege. Infolgedessen ergibt sich eine asymmetrische Reaktion, wobei mit einem Anstieg der Volatilität eine stärkere Reaktion der Aktienrenditen verbunden ist als im Fall eines Rückgangs.[49] GARCH-Modelle können den Leverage-Effekt nicht berücksichtigen, da die quadrierte Tagesrendite stets unkorreliert mit der Rendite der Vorperiode ist. Aus diesem Grund wurden asymmetrische Erweiterungen wie das GJR-GARCH bzw. (Treshhold) TGARCH, das (Nonlinear Asymmetric) NAGARCH oder das (Exponential) EGARCH-Modell entwickelt, um den Leverage-Effekt empirisch zu modellieren.[50]

[48] Vgl. Schmid / Trede (2006), S. 14.
[49] Vgl. Mauer / Nastansky (2024), S. 11f.
[50] Vgl. Schmid / Trede (2006), S. 182; Schmitt (2012), S. 286ff.

3.3 EGARCH-Modell

Um asymmetrische Effekte in der Volatilitätsentwicklung zu erfassen, wurde von Nelson (1991) das Exponential-GARCH-Modell (EGARCH) entwickelt. Eine zentrale Eigenschaft dieses Modells ist die logarithmische Transformation der bedingten Varianz, wodurch die üblichen Nicht-Negativitätsbedingungen entfallen. Dies vereinfacht die Schätzung des Modells und erhöht die Flexibilität der Parametrisierung. Formal lässt sich das EGARCH (*p*,*q*)-Modell wie folgt darstellen:[51]

$$\ln \sigma_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i \, g(z_{t-i}) + \sum_{j=1}^{p} \beta_j \ln \sigma_{t-j}^2 \tag{3.9}$$

$$g(z_t) = \theta \left[\left| \frac{\varepsilon_{t-i}}{\sigma_{t-i}} \right| - \mathrm{E}\left(\left| \frac{\varepsilon_{t-i}}{\sigma_{t-i}} \right| \right) \right] + \gamma \frac{\varepsilon_{t-i}}{\sigma_{t-i}} \tag{3.10}$$

wobei $g(z_t)$ die Auswirkungen zufälliger Kursbewegungen (Schocks) auf die bedingte Varianz berücksichtig und $\frac{\varepsilon_{t-i}}{\sigma_{t-i}}$ bilden die standardisierten Fehlerterme ab. Der Ausdruck $\theta \left[\left| \frac{\varepsilon_{t-i}}{\sigma_{t-j}} \right| - \mathrm{E}\left(\left| \frac{\varepsilon_{t-i}}{\sigma_{t-j}} \right| \right) \right]$ erfasst die Größenabhängigkeit von den *i* Perioden zurückliegenden Schocks. Ein größerer Wert für θ zeigt einen höheren ARCH-Effekt an.[52] Der zweite Term, $\gamma \frac{\varepsilon_{t-i}}{\sigma_{t-i}}$, repräsentiert die asymmetrische Reaktion der Volatilität auf positive und negative Schocks und gibt damit den Leverage-Effekt wieder. Ist $\gamma = 0$, reagiert die Volatilität symmetrisch auf beide Schockrichtungen. Wenn $-1 < \gamma < 0$ gilt, erhöhen negative Schocks die Volatilität stärker als positive Nachrichten. Werte von $\gamma < -1$ führen dazu, dass positive Schocks die Volatilität reduzieren; während negative sie verstärken. Allerdings widersprecht dieser Fall der ökonomischen Intuition.[53]

Im Vergleich zum klassischen GARCH-Modell bietet das EGARCH-Modell folgende Vorteile: Zum einen stellt die logarithmische Transformation sicher, dass die Varianz stets positiv bleibt, ohne dass explizite Parameterrestriktionen notwendig sind. Zum anderen erlaubt das Modell eine differenzierte Reaktion der Volatilität auf positive und negative Marktbewegungen, was insbesondere für Finanzmarktdaten eine realistischere Abbildung ermöglicht. Als Nachteil wurde festgestellt, dass EGARCH-Modelle den Effekt großer Schocks auf die Volatilität tendenziell übergewichten und daraus schlechtere Anpassungen resultieren können.[54]

---

[51] Vgl. Nelson (1991), 350f.
[52] Für empirische Anwendungen kann vereinfachend angenommen werden, dass der Koeffizient $\theta = 1$ ist.
[53] Vgl. Schmitt (2012), S. 286.
[54] Vgl. ebenda, S. 288.

3.4 GRJ-GARCH-Modell

Während das EGARCH-Modell die Asymmetrie der Volatilitätsreaktion über eine logarithmische Transformation der Varianz erfasst, existieren alternative Modellansätze, die asymmetrischen Effekte direkt in die Varianzgleichung integrieren. Die Idee von Schwellenwert (Treshold) Modellen ist eine Unterteilung der zufälligen Kursbewegungen (Schocks) in disjunkte Intervalle und eine stückweise lineare Funktion entweder für die bedingte Standardabweichung wie im TGARCH nach Zakoian (1994) oder für die bedingte Varianz nach Glosten-Jagannathan-Runkle (1993) mit dem GJR-GARCH anzupassen.[55] Im Fall von nur zwei Intervallen (Unterscheidung zwischen positiven und negativen Schocks auf die Volatilität) wird der Schwellenwert null angenommen.

Um die asymmetrische Dynamik zu erfassen, wird die bedingte Varianz gemäß (3.4) um eine Indikatorfunktion $\boldsymbol{I_{t-i}}$ ergänzt. Dieser wird jedoch nur bei negativen Schocks aktiviert. Formal ergibt sich die folgende Modellgleichung:[56]

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} (\alpha_i + \gamma_i \boldsymbol{I_{t-i}}) \varepsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2 \tag{3.11}$$

$$\text{mit} \quad \boldsymbol{I_{t-i}} = \begin{cases} 0 & \varepsilon_{t-1} \geq 0 \\ 1 & \varepsilon_{t-1} < 0 \end{cases} \tag{3.12}$$

wobei die Nichtnegativität durch die Bedingungen α_0 , $\alpha_i + \gamma > 0$ und $\beta_j > 0$ gewährleistet wird. Der Parameter γ beschreibt die zusätzliche Volatilitätsreaktion auf negative Schocks.[57] Falls $\gamma = 0$ ist, reduziert sich das Modell auf das (symmetrische) GARCH(p,q)-Modell. Das zentrale Merkmal des GJR-GARCH-Modells ist die explizite Trennung der Schockarten. Positive Kursänderungen beeinflussen die Volatilität weniger stark als negative Schocks, wodurch bei $\gamma > 0$ der Leverage-Effekt eintritt und dieser gegen das $(1 - \alpha)$-Quantil der Standardnormalverteilung getestet werden kann.

Grundsätzlich könnte der abrupte Übergang zwischen zwei Renditeregimes als Folge des Überschreitens eines Schwellenwerts auch durch einen glatteren Übergang ersetzet werden. Hierfür müsste die Indikatorfunktion durch eine beliebige stetige Funktion ersetzte werden, die für kleine Werte ε_{t-i} der unerwarteten Kursausschläge des Finanzinstruments gegen null strebt und für große Werte gegen Eins geht.[58]

---

[55] Vgl. Schmid / Trede (2006), S. 185.
[56] Vgl. Brooks (2008), S. 405.
[57] Vgl. Schmitt (2012), S. 288.
[58] Vgl. Franke et al. (2003), S. 231.

3.5 Maximum-Likelihood Schätzung von GARCH-Modellen

Die Parameter der (G)ARCH-Modelle und deren asymmetrische Erweiterungen werden üblicherweise mit der Maximum-Likelihood-Methode (ML) geschätzt. Die ML-Schätzung versucht dabei denjenigen Wert aus dem zulässigen Parameterraum als Schätzer auszuwählen, unter denen die gegebene Stichprobe die größte Wahrscheinlichkeit (bzw. Wahrscheinlichkeitsdichte) besitzt.[59]

Grundsätzlich basiert die Methode auf der Maximierung der Likelihood-Funktion L, die die Wahrscheinlichkeit der beobachteten Daten als Funktion der Modellparameter darstellt. Hierbei wird davon ausgegangen, dass die Parameter variabel sind und die empirischen Renditewerte feststehen. Dabei ist die Annahme einer Verteilung für die Methode unerlässlich. An dieser Stelle wird von einer bedingten Normalverteilung von ϵ_t ausgegangen, wenngleich auch eine t-Verteilung mit einer geringen Zahl von Freiheitsgraden (stärker besetzte Enden) möglich wäre. Bereits ab 100 Beobachtungen unterscheiden sich die Quantile der t-Verteilung praktisch kaum von denen der (Standard-)Normalverteilung). In der Praxis wird zudem mit dem natürlichen Logarithmus der Likelihood Funktion (Log-Likelihood) gearbeitet, da dieser mathematisch einfacher zu handhaben ist. Die Log-Likelihood Funktion des GARCH(p,q)-Prozesses ergibt sich wie folgt:[60]

$$\ln L(\theta) = \sum_{t=1}^{T} l_t = -\frac{T}{2}\ln(2\pi) - \frac{1}{2}\sum_{t=1}^{T}\left(\ln(\sigma_t^2) + \frac{\epsilon_t^2}{\sigma_t^2}\right) \tag{3.13}$$

mit $\theta = (\alpha_0, \alpha_1, \ldots, \alpha_p, \beta_1, \ldots, \beta_q)$ und $\sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2$ sowie die logarithmierte bedingte Dichte l_t und die Anzahl der Beobachtungen T.

Ziel des ML-Verfahrens ist die Maximierung von (3.13) in Abhängigkeit des Parametervektors θ. Für die Schätzung sind die partiellen Ableitungen der Funktion nach den Parametern sowie die Hesse-Matrix zu bestimmen. Aufgrund der rekursiven Struktur der bedingten Varianzen ist dies jedoch sehr schwierig. Eine analytische Lösung ist folglich meist nicht verfügbar, weshalb numerische Optimierungsalgorithmen wie der Berndt-Hall-Hall-Hausman (BHHH)-Algorithmus) oder der Sequential-Least-Squares-Programming (SLSQP)-Algorithmus Anwendung finden.[61] Hierbei empfiehlt es sich als Startwerte möglichst kleine Werte von p und q zu wählen, da die Einhaltung der Stationaritätsbedingung in der Praxis schwierig sein kann.

---

[59] Vgl. Pesaran (2015), S. 198f.
[60] Vgl. Schmitt (2012), S. 282.
[61] Vgl. Bollerslev (1986), S. 315f.; Schmitt (2012), S. 283.

Damit die Maximum-Likelihood-Schätzer mit ihren vorteilhaften Eigenschaften bestimmt werden können, müssen die folgenden Regularitätsbedingungen erfüllt sein: [62]

- Erstens müssen die ersten drei Ableitungen des Log-Likelihood-Funktion bezüglich θ stetig und endlich sein. Damit wird sichergestellt, dass eine Taylor-Reihen-Approximation existiert und die Varianz der Ableitungen endlich ist.
- Zweitens müssen die Bedingungen, die notwendig sind, um die Erwartungswerte der ersten und zweiten Ableitung von (3.13) zu erhalten, erfüllt sein.
- Drittens müssen alle Werte von θ für die 3. Ableitung kleiner sein als eine Funktion, die einen endlichen Erwartungswert hat. Diese Bedingung ermöglicht es, die Taylor-Reihe abzuschneiden.

Sind die obigen Bedingungen erfüllt, weisen die ML-Parameterschätzer die folgenden Eigenschaften auf: [63]

- Die Schätzer sind konsistent: Sie konvergieren gegen die wahren (unbekannten) Parameter, wenn die Stichprobengröße gegen unendlich geht. Somit wird mit zunehmender Anzahl an Beobachtungen (länge der Zeitreihen) die Schätzung immer genauer.
- Die Schätzer sind asymptotisch (gemeinsam) normalverteilt: Diese Eigenschaft ermöglicht es, Konfidenzintervalle für die Parameter zu berechnen und Hypothesentests durchzuführen.
- Die Schätzer sind asymptotisch effizient: Sie weisen die geringste asymptotische Varianz unter allen konsistenten Schätzern auf.
- Die Schätzer sind invariant: Diese Eigenschaft stellt sicher, dass bei einer Transformation der Parameter durch stetig differenzierbare Funktionen die Schätzeigenschaften der transformierten Funktion erhalten bleiben.

Insgesamt ermöglichen diese stochastischen Eigenschaften der ML-Schätzung eine präzise und zuverlässige Bestimmung der Parameter des GARCH(p,q)-Modells, was insbesondere in Anwendungen mit begrenzten Datenmengen entscheidend für die genaue Modellierung und Prognose der Volatilität ist. Falls die Annahme der Normalverteilung auf die standardisierten Residuen v_t nicht zutrifft, liefert die ML-Methode zumindest konsistente Parameterschätzer und wird als Quasi-Maximum-Likelihood (QML)-Schätzer bezeichnet.

[62] Vgl. Greene (2019), S. 583.
[63] Vgl. Schmid / Trede (2006), S. 173; Auer (2023), S. 515ff.; Greene (2019), S. 582.

4 Anwendung von Deep Learning in der Volatilitätsprognose

4.1 Grundlagen der neuronalen Netzwerke

Künstliche Neuronale Netze (KNN) sind vom biologischen Nervensystem inspiriert. McCulloch und Pitts (1943) führten als Erste das Neuronenmodell als mathematisches Modell zur Abbildung der neuronalen Aktivität des menschlichen Gehirns ein. Wenngleich das Interesse an den KNN in wirtschaftswissenschaftlichen Anwendungen zunächst gering war, so erleben sie aktuell eine Renaissance. Im Gegensatz zu traditionellen zeitreihenökonometrischen Modellen bieten Neuronale Netze mehrere Vorteile.

Klassische ökonometrische Verfahren basieren auf zahlreichen Annahmen. Sie sind zudem parametrisch und setzen eine spezifische Form voraus. KNN hingegen sind nicht-parametrisch und können komplexe, nicht-lineare Beziehungen lernen − ohne a priori Annahmen über die Beziehung zwischen den Variablen tätigen zu müssen. Diese Eigenschaft macht KNN besonders interessant, wenn die zugrundeliegenden Annahmen der ökonometrischen Modelle nicht adäquat der Realität entsprechen. Überdies kann die Struktur der Netze beliebig verändert werden, um ein breites Spektrum von ökonomischen Prozessen zu approximieren. Somit sind Neuronale Netze auch für die Volatilitätsprognose geeignet.[64]

Das grundlegende Bauelement eines jeden KNN ist das Perzeptron, d. h., ein einfaches mathematisches Modell der Funktionsweise einer menschlichen Nervenzelle (Neuron). Ein Perzeptron erhält eine oder mehrere Eingabenvariablen x_i, die jeweils mit Gewichten w_i versehen und mit einem Bias b zu einer Summe kombiniert werden.[65] Diese Summe wird durch eine Aktivierungsfunktion $f(.)$ geleitet, die entscheidet, ob das Neuron aktiviert wird oder nicht. Die Ausgangsvariable y durch die folgende Funktion $y = f(\sum_{i=1}^{n} w_i x_i + b)$ dargestellt werden. In dieser entspricht der Bias der Konstante in einem Regressionsmodell.

Das menschliche Gehirn ist eine Ansammlung vollständig verbundener Neuronen. Die Signalübertragung kann dabei grob in drei Schritte unterteilt werden: (1) Zunächst empfangen die Dendriten chemische Signale von benachbarten Neuronen. (2) Der Zellkörper integriert anschließend die empfangenen Signale und erzeugt ein Aktionspotenzial, wenn ein Schwellenwert überschritten wird. (3) Das Aktionspotenzial wird entlang des Axons weitergeleitet und führt zur Freisetzung von Neurotransmittern an den Synapsen. Analog erfolgt die Verknüpfung von Neuronen in einem (vorwärtsgerichteten) KNN (siehe Abb. 8).[66]

---

[64] Vgl. Vollmer et al. (1994), S. 41f.
[65] Vgl. Nielsen (2015), S. 3ff.
[66] Vgl. Becker-Carus / Wendt (2017), S. 34ff.

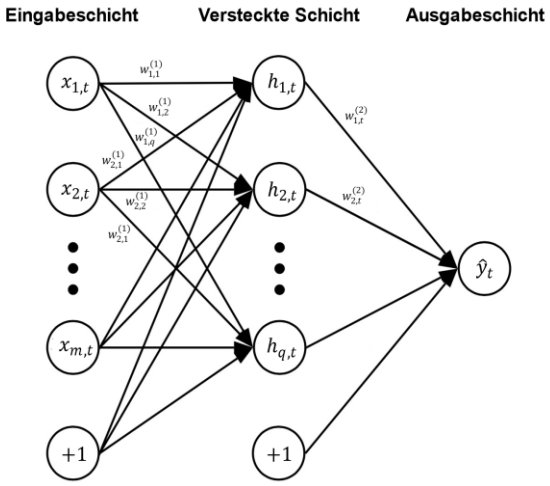

Abb. 8: Feedforward Neural Network Diagramm
Quelle: Zhang et al. (2023), S. 169 (Eigene Darstellung)

Die Eingabeschicht (Input Layer) beinhaltet sämtliche exogenen Variablen $x_{m,t}$. Soll die Volatilität zu verschiedenen Zeitpunkten t vorhergesagt werden, jeweils auf der Grundlage der Volatilität und der logarithmischen Rendite der vorherigen Zeitpunkte, so ist die Anzahl der Merkmale $m = 2$ und jede Prognose $\hat{y}_t$ wird durch die vorherigen Werte $\mathbf{x} = \left[x_{1,t}, x_{2,t}, \ldots, x_{m,t}\right]^T$ definiert. Daraus folgt eine versteckte Schicht (Hidden Layer) mit versteckten Neuronen, deren Ausgaben als versteckte Repräsentationen $\mathbf{h} = \left[h_{1,t}, h_{2,t}, \ldots, h_{q,t}\right]^T$ bezeichnet werden. Diese werden beispielsweise für eine versteckte Schicht in (4.1) dargestellt. Jede Schicht ist somit über Kanten mit einer dazugehörigen Gewichtsmatrix $\mathbf{W}$ und einem Biasvektor $\mathbf{b}$ verbunden. Letztlich gibt es die Ausgabeschicht (Output Layer), die die Volatilitätsprognose ausgibt (siehe (4.2)). $f(.)$ und $g(.)$ bilden zwei verschiedene Aktivierungsfunktionen ab:[67]

$$\mathbf{h} = f\left(\mathbf{x}\mathbf{W}^{(1)} + \mathbf{b}^{(1)}\right) \tag{4.1}$$

$$\hat{\mathbf{y}}_t = g\left(\mathbf{h}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}\right) \tag{4.2}$$

$f(.)$ und $g(.)$ können aus einer Vielzahl von Aktivierungsfunktionen gewählt werden. Die relevantesten sind die Sigmoidfunktion (σ), die hyperbolische Tangensfunktion (Tanh) und die lineare Funktion. Die Sigmoidfunktion (auch als logistische Funktion bezeichnet) hat einen Wertebereich zwischen $(0; 1)$ und die Tanh-Funktion von $(-1; 1)$. Die lineare Funktion kann für Regressionsprobleme herangezogen werden.[68] Im Fall von nicht-negativ definierten reellen Eingabewerten für $\mathbf{x}$ ist die Rectified Linear Unit (ReLU)-Funktion anzusetzen.

[67] Vgl. Zhang et al. (2023), S. 169ff.
[68] Vgl. ebenda, S. 172f.

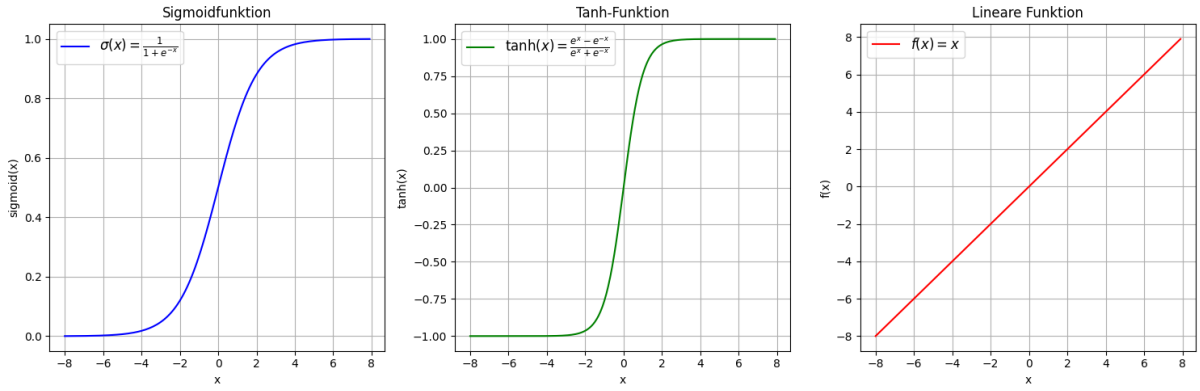

Abb. 9: Aktivierungsfunktionen
Quelle: Ganguly (2021), S. 106 (Eigene Darstellung).

Die Wahl der Aktivierungsfunktion spielt eine wichtige Rolle für die Architektur eines neuronalen Netzes. Wären alle Aktivierungsfunktionen linear, könnte das KNN auf ein lineares Regressionsmodell reduziert werden. Werden hingegen nicht-lineare Funktionen wie Sigmoid oder Tanh verwendet, können die Netzwerke komplexe funktionale Zusammenhänge entdecken.[69]

4.2 Gradient Descent und Backpropagation

Ähnlich wie das menschliche Gehirn lernen auch Neuronale Netze durch Erfahrungen. Beide Systeme haben die Fähigkeit, sich als Reaktion auf Fehler anzupassen und ihr Verhalten zu optimieren. Während das menschliche Gehirn durch hochkomplexe neuronale und biochemische Prozesse lernt, die noch nicht vollständig verstanden sind, lernen die KNN durch eine vereinfachte, aber effektive Nachbildung dieses Lernprozesses.[70]

Um den Begriff „Fehler“ im Kontext neuronaler Netze zu präzisieren, ist es notwendig, eine Verlustfunktion L einzuführen. Für die Berechnung des Fehlers und damit der Bewertung der Prognosegüte von Regressionsproblemen wie die Vorhersage der Volatilität kann u.a. der mittlere quadratische Fehler (Mean Squared Error, MSE) verwendet werden. Folglich berechnet die Verlustfunktion den mittleren quadratischen Abstand zwischen den tatsächlichen und den vorhergesagten Werten:[71]

$$L = \frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{y}_t)^2 \tag{4.3}$$

[69] Vgl. Ganguly (2021), S. 105f.
[70] Vgl. Becker-Carus / Wendt (2017), S. 293ff.
[71] Vgl. Zhang et al. (2023), S. 84f.

Das Ziel des Lernalgorithmus besteht darin, die Gewichte (entsprechen den Koeffizienten eines Regressionsmodells) so anzupassen, dass die Verlustfunktion minimiert wird. Ein gebräuchlicher Ansatz zur Lösung dieses numerischen Optimierungsproblems ist der Gradientenabstieg (Gradient Descent). [72] Bei dieser Methode wird iterativ ein Minimum einer nichtlinearen, differenzierbaren Funktion gesucht, indem bei jeder Iteration ein Schritt in die entgegengesetzte Richtung des Gradienten der Funktion an der aktuellen Stelle durchgeführt wird.[73] Um das Verfahren zu veranschaulichen, wird eine konkave Funktion f einer Veränderlichen x betrachtet. Wenn $f'(x) > 0$ an einem bestimmten Punkt gilt, folgt daraus, dass $f(x)$ näherungsweise zunimmt, wenn x ebenfalls steigt. Durch Verringerung von x wird ein lokales Minimum erreicht.[74] Umgekehrt, wenn $f'(x) < 0$ ist, muss x erhöht werden, damit ein lokales Minimum gefunden wird.

Bei der Suche nach einem Minimum einer mehrdimensionalen Funktion wird das gleiche Verfahren angewendet. Allerdings wird nicht nur die Variable x betrachtet, sondern alle partiellen Ableitungen, um die Bewegung in Bezug auf mehrere Dimensionen anzupassen. Der Vektor, der alle partiellen Ableitungen $\frac{\partial f}{\partial x}$ mit $\mathbf{x} = [x_1, x_2, \dots, x_t]^T$ enthält, wird als Gradient bezeichnet und ist als $\nabla f(x) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_t}\right]^T$ definiert.[75]

Gemäß des Gradient Descent werden nach jeder Iteration die Gewichte des neuronalen Netzes wie folgt angepasst:[76]

$$\mathbf{W}^{(1)'} = \mathbf{W}^{(1)} - \eta \nabla L\left(\mathbf{W}^{(1)}\right) \tag{4.4}$$

$$\mathbf{W}^{(2)'} = \mathbf{W}^{(2)} - \eta \nabla L\left(\mathbf{W}^{(2)}\right) \tag{4.5}$$

Hierbei sind $\mathbf{W^{(1)}}$ und $\mathbf{W^{(2)}}$ Matrizen, die alle Gewichte zwischen den ersten und letzten beiden Schichten enthalten. Der Parameter η wird als Schrittweite bzw. Lernrate (Learning Rate) bezeichnet und steuert die Größe der Gewichtsänderung. Die Lernrate hat einen Einfluss auf die Konvergenz des Lernalgorithmus: Ist sie zu hoch ist, kann der Algorithmus das Tal verfehlen und geht über das Minimum hinaus. Ist die Lernrate hingegen zu niedrig, dauert es länger, bis das Netzwerk konvergiert (siehe Abb. 10).[77]

---

[72] Der Gradient beschreibt die Richtung des steilsten Anstiegs von f. Umgekehrt beschreibt $-\nabla f$ die Richtung des steilsten Abstiegs.
[73] Vgl. Zhang et al. (2023), S. 468f.
[74] Vgl. Sydsaeter et al. (2018), S. 214ff.
[75] Vgl. Zhang et al. (2023), S. 468f.
[76] Vgl. ebenda, S. 486.
[77] Vgl. Goodfellow et al. (2016), S. 83f.

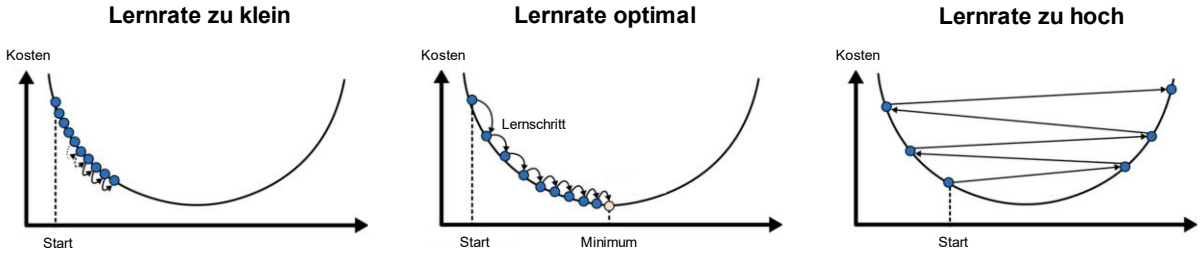

Abb. 10: Auswahl der Lernrate
Quelle: Géron (2023), S. 206f. (Eigene Darstellung).

Ein weiteres Problem besteht darin, dass der Gradient Descent-Algorithmus nur lokale Minima bestimmen kann, d. h., dass er möglicherweise nicht das globale Minimum der Verlustfunktion findet. In einem hochdimensionalen Raum, wie es bei neuronalen Netzen der Fall ist, können viele lokale Minima existieren. Der Algorithmus kann in einem dieser lokalen Minima „stecken bleiben", was dazu führt, dass das Netzwerk suboptimale Lösungen erlernt. Dies kann die Generalisierbarkeit des Modells beeinträchtigen und z. B. zu geringerer Prognosegüte auf unbekannten Daten (zukünftige Werte) führen.[78]

Mit dem Backpropagation-Algorithmus können die Gradienten der Verlustfunktion in Bezug auf die Modellparameter und somit die Gewichte effizient und strukturiert für mehrere Schichten bestimmt werden. In einem ersten Schritt (Forward Pass) werden die Eingaben durch das Netz geleitet, um die Ausgaben zu berechnen.[79] Dabei werden die Aktivierungen und Zwischenergebnisse für jeden Schritt gespeichert, die später für die Rückwärtsberechnung der Gradienten benötigt werden. Im zweiten Schritt (Backward Pass) wird bei der Ausgabeschicht begonnen und Schritt für Schritt der Gradient der Verlustfunktion berechnet. Diese Gradienten werden dann verwendet, um die Gewichte des Netzes zu aktualisieren (siehe (4.4) und (4.5)). Durch die schrittweise Anpassung der Gewichte wird das KNN in die Lage versetzt, die Fehler zu reduzieren und die Vorhersagen für den Output zu verbessern.

Obwohl Backpropagation es ermöglicht, effektiv partielle Ableitungen in Bezug auf die Modellparameter zu berechnen; können große Datensätze den Lernprozess immer noch unpraktikabel gestalten. Grund dafür ist, dass der Gradientenabstieg den Gradienten nach jeder Epoche, d. h. nachdem T Datenpunkte durch das neuronale Netz gelaufen sind, berechnet. Die partiellen Ableitungen der Verlustfunktion in Bezug auf die Modellparameter enthalten dann alle T Datenpunkte. Folglich wird die Berechnung mit zunehmender Größe des Datensatzes immer komplexer.[80]

[78] Vgl. Zhang et al. (2023), S. 485.
[79] Vgl. Géron (2023), S. 401f.
[80] Vgl. Goodfellow et al. (2016), S. 151f.

Eine Lösung für dieses Problem bietet die Schätzung der Gradienten mithilfe des Stochastische Gradientenabstieg (Stochastic Gradient Descent, SGD). Gemäß dieser Näherungsmethode werden die Gewichte nicht nach jeder Epoche, sondern nach einem einzelnen Datenpunkt angepasst. Demzufolge ist der Rechenaufwand für eine einzelne Iteration deutlich geringer. Da die Varianz zwischen den einzelnen Gradienten hoch ist, kann der SGD-Algorithmus leichter aus einem Sattelpunkt oder lokalem Minimum herausgelangen. Da der Gradient nur aus einem Datenpunkt geschätzt wird, sind die Schritte in Richtung des Minimums der Verlustfunktion zwar weniger präzise (hohe Fehlervarianz) und führen nicht zwangsläufig dazu, dass der Fehler abnimmt. Auch besteht die Gefahr einer schlechten Konvergenz zum Minimum, da Gradient nur geschätzt wird (mehr Zickzack). Langfristig gelingt es der Methode jedoch, den durchschnittlichen Gesamtfehler zu minimieren.[81]

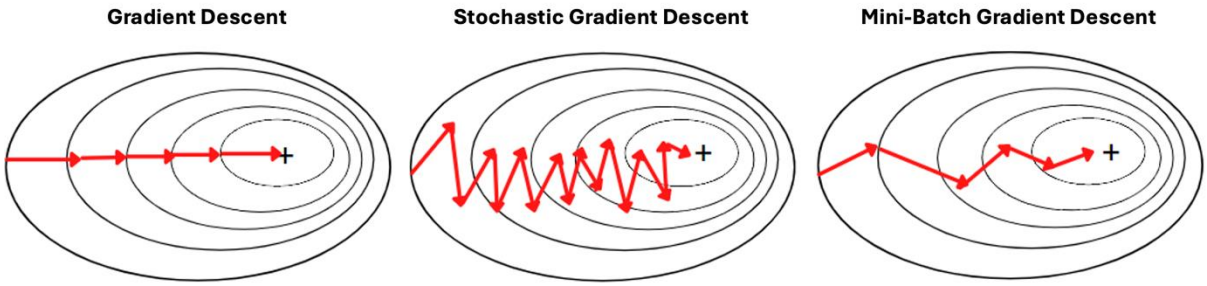

Abb. 11: Arten des Gradientenabstiegs
Quelle: Wagh (2022), Absatz 3 (Eigene Datstellung).

Eine weitere Lösung bietet eine Mischung aus Gradientenabstieg (Gradient Descent) und SGD, die als Mini-Batch-Gradient-Descent bezeichnet wird. Diese Methode ähnelt dem Gradientenabstieg; allerdings wird der Gradient nicht über alle Datenpunkte im Trainingsdatensatz berechnet, sondern über eine Teilmenge (Batch) davon. Folglich reduziert der Mini-Batch-Gradient-Descent die Varianz im Gradientenpfad (wie beim Standardgradientenabstieg) und erhöht die Geschwindigkeit bzw. Effizienz (wie beim SGD). Der Stochastische Gradientenabstieg ist somit ein Sonderfall des Mini-Batch Gradientenabstiegs mit einer Batchgröße von 1.[82]

Der Backpropagation-Algorithmus ist beendet, wenn das zuvor definierte Abbruchkriterium erfüllt ist. Führt das Training hingegen zu einem unbefriedigenden Ergebnis, können entweder die Anzahl der Neuronen, die Netztopologie, das Abbruchkriterium oder die Lernrate angepasst werden.

[81] Vgl. Géron (2023), S. 212f.
[82] Vgl. ebenda, S. 216f.

4.3 Recurrent Neural Networks

Ein neuronales Netzwerk, bei dem die Signalübertragung von der Eingabeschicht einmal durch das Netz zur Ausgabeschicht führt, wird als Feedforward Neural Network (vorwärtsgekoppelte Neuronale Netze, FNN) bezeichnet. Im Gegensatz dazu ermöglichen die von Elman (1990) entwickelten Recurrent Neural Networks (rückgekoppelte Neuronale Netze, RNNs) eine Rückkopplung innerhalb des Netzwerks. Dabei werden Informationen aus vorherigen Zeitschritten gespeichert und können zukünftige Berechnungen beeinflussen, wodurch das Netzwerk zeitliche Abhängigkeiten erfasst (Rückkopplungsmechanismen). Traditionelle FNNs werden auch als statische Neuronale Netze bezeichnet, da sie den Output basierend auf unabhängigen Merkmalsgruppen fester Länge vorhersagen und die sequenzielle Reihenfolge der Beobachtungen ignorieren. Neue Inputs werden isoliert betrachtet, ohne Berücksichtigung vorheriger Inputs. Daher sind FNNs für sequenzielle Daten wie Zeitreihen ungeeignet. [83]

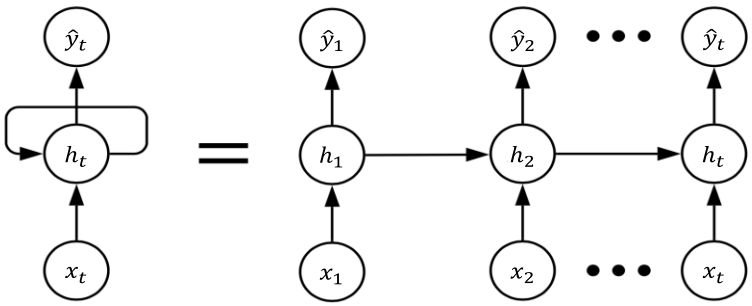

Abb. 12: Entfaltetes Diagramm eines Recurrent Neural Networks
Quelle: Zhang et al. (2023), S. 326 (Eigene Darstellung).

Im Gegensatz zu FNNs ermöglichen RNNs die Weiterleitung von Daten − nicht nur von der Eingabeschicht zur Ausgabeschicht, sondern auch durch versteckte Schichten früherer Zeitpunkte. Im Vergleich zur Abb. 8 repräsentieren die Kreise in der RNN-Darstellung Abb. 12 ganze Schichten. $\mathbf{x}_t$ und $\mathbf{h}_t$ sind Vektoren, die alle Inputs (z. B. logarithmische Renditen und historische Volatilitäten) sowie die versteckten Zustände der t-ten Beobachtung enthalten. Mit Blick auf die rechte Seite von Abb. 12 wird deutlich, dass ein RNN als eine Aneinanderreihung mehrerer Kopien von FNNs gesehen werden kann, die jeweils eine Nachricht an den Nachfolger übermitteln. Diese Nachricht, dargestellt durch $\mathbf{h}_t$, wird im Kontext von RNNs als versteckter Zustand bezeichnet. Auf diese Weise können RNNs sich an frühere Informationen erinnern und Zeitabhängigkeiten in den Daten erkennen.[84]

[83] Vgl. Haykin (2009), S. 21ff.
[84] Vgl. Elman (1990), S. 182f.

Unter Beibehaltung der für FNNs verwendeten Notation kann der Output eines RNNs wie folgt dargestellt werden:[85]

$$h_t = f\left(\mathbf{W}^{(1)}\mathbf{x}_t + \mathbf{U}^{(1)}\mathbf{h}_{t-1} + \mathbf{b}^{(1)}\right) \tag{4.6}$$

$$\hat{y}_t = g\left(\mathbf{W}^{(2)}\mathbf{h}_t + \mathbf{b}^{(2)}\right) \tag{4.7}$$

Wobei $\mathbf{U}^{(1)}$ eine Matrix ist, die die Gewichte für alle Verbindungen zwischen $\mathbf{h}_{t-1}$ und $\mathbf{h}_t$ enthält. RNNs können sodann als FNNs mit zusätzlichen Inputs betrachtet werden.[86]

Aufgrund dieser Ähnlichkeiten werden RNNs auch mittels Backpropagation trainiert. Der einzige Unterschied besteht darin, dass der Fehler nicht nur durch die Schichten, sondern auch durch die Zeit propagiert wird. Folglich wird der Algorithmus für die RNNs als Backpropagation Through Time (BPTT) bezeichnet. Obwohl die BPTT keine rechnerischen Probleme verursacht, leidet sie unter dem Problem der verschwindenden (Vanishing) oder explodierenden (Exploding) Gradienten.[87]

Die Berechnung des Gradienten beinhaltet die Multiplikation zahlreicher partieller Ableitungen. Bei langen Sequenzen kann das Produkt vieler Werte im Bereich von -1 bis 1 dazu führen, dass der Gradient exponentiell abnimmt und somit gegen Null tendiert. Umgekehrt, wenn viele partielle Ableitungen absolute Werte größer als 1 annehmen, wird der Gradient explodieren. Verschwindende Gradienten bewirken, dass das neuronale Netz die Gewichte nicht mehr anpasst, wodurch der Lernprozess stoppt. Explodierende Gradienten hingegen führen zu übermäßigen Gewichtsanpassungen, was zu instabilen Gewichtsmatrizen führt. Beide Probleme verhindern, dass RNNs langfristige Abhängigkeiten erfassen können.[88]

4.4 Long Short-Term Memory Netzwerke

Um die mit den RNNs verbundenen Gradientenprobleme der Instabilität über die Zeit zu überwinden, haben Hochreiter und Schmidhuber (1997) eine spezielle Art von Recurrent Neural Networks eingeführt, das als Long Short-Term Memory (LSTM) bekannt ist und erfolgreich bei der Erkennung von Langzeitabhängigkeiten eingesetzt werden kann. Um RNN verwenden zu können, werden nicht nur die bereits beschriebenen Neuronen verwendet, sondern in den meisten Fällen sogenannte LSTM-Zellen (Lange Kurzzeit-Gedächtnis Zellen).[89]

[85] Vgl. Zhang et al. (2023), S. 349f.
[86] Vgl. ebenda, S. 349ff.
[87] Vgl. ebenda, S. 367ff.
[88] Vgl. ebenda, S. 367.
[89] Vgl. Hochreiter / Schmidhuber (1997), S. 1735ff.

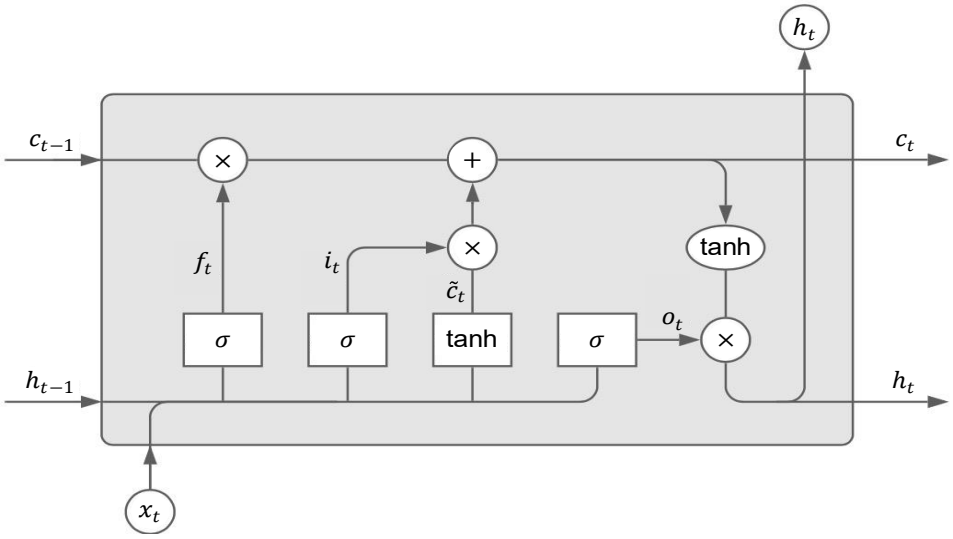

Abb. 13: Long Short-Term Memory-Diagramm
Quelle: Zhang et al. (2023), S. 373 (Eigene Darstellung).

Bei den LSTM-Zellen wird im Vergleich zu den Neuronen der innere Aufbau durch einen Zellzustand erweitert. Die Eingabegrößen werden hierbei mit einer Aktivierungsfunktion aktiviert und anschließend mit dem Zellzustand der LSTM-Zelle verrechnet. Der Zellzustand kann sich mit jedem Zeitschritt ändern, da dieser abhängig von den vorangegangenen Eingabewerten ist. Im Vergleich zu herkömmlichen RNNs sind LSTMs damit in der Lage, zu jedem Zeitpunkt des Trainingsprozesses auszuwählen, welche Informationen aus der Vergangenheit vergessen, welche neuen Informationen hinzugefügt und welche ausgegeben werden sollen. Dies wird durch drei verschiedene Gates erreicht: das Forget Gate, das Input Gate und das Output Gate, die die Informationen im sogenannten Cell State regulieren: Der Cell State-Vektor $\mathbf{c}_t$ fungiert als Langzeitspeicher, der Informationen auf eine kontrollierte Weise speichert und weiterleitet. Die Gates sind u. a. abhängig von der versteckten Schicht, die als Kurzzeitspeicher fungiert, neue Informationen verarbeitet und temporär speichern kann. [90]

Für ein besseres Verständnis über die Funktionsweise des LSTMs wird in Abb. 13 das LSTM-Modul veranschaulicht. Da die Veränderungen gegenüber traditionellen RNNs nur die versteckte Schicht betrifft, wird die Ausgabeschicht nicht dargestellt. Informationen fließen sowohl vertikal von der Eingabe- zur versteckten Schicht als auch horizontal von der vorherigen zur aktuellen versteckten Schicht. Rechteckige Formen repräsentieren Schichten, wobei der Text darin die angewandte Aktivierungsfunktion angibt. Kreise und Ellipsen innerhalb des LSTM-Moduls stehen für elementweise Operationen, und beschriftete Linien bezeichnen die übergebenen Vektoren.[91]

[90] Vgl. Géron (2023), S. 749f.
[91] Vgl. Zhang et al. (2023), S. 373.

Beginnend mit dem Forget Gate (links) werden $\mathbf{x}_t$ und $\mathbf{h}_{t-1}$ durchgeleitet, um den Vektor $\mathbf{f}_t$ zu bestimmen. Da die Sigmoidfunktion nur Werte im Bereich von 0 und 1 ausgibt, bedeuten Elemente von $\mathbf{f}_t$, die einem Wert von 1 entsprechen, „alle Informationen behalten", während Werte von 0 „alle Informationen vergessen" bedeuten. Die Multiplikation des Vektors $\mathbf{f}_t$ mit dem vorherigen Cell State $\mathbf{c}_{t-1}$ bestimmt somit, ob der aktuelle Wert des Langzeitspeichers beibehalten oder gelöscht werden soll. Daraus folgend können vergangene Informationen auch teilweise übernommen werden, wenn der entsprechende Wert von $\mathbf{f}_t$ zwischen 0 und 1 liegt. Der Vektor des Forget Gates wird wie folgt bestimmt:[92]

$$\mathbf{f}_t = \sigma\left(\mathbf{W}^{(f)}\mathbf{x}_t + \mathbf{U}^{(f)}\mathbf{h}_{t-1} + \mathbf{b}^{(f)}\right) \tag{4.8}$$

Der nächste Schritt besteht darin, zu entscheiden, welche neuen Informationen in den Cell State aufgenommen werden sollen. Dies geschieht durch die folgenden zwei Schichten: Input Gate $\mathbf{i}_t$ und Input Node $\tilde{\mathbf{c}}_t$. Das Input Gate betrachtet die aktuellen Eingaben und den vergangenen versteckten Zustand. Deren gewichtete Summe wird durch die Sigmoidfunktion geleitet, um zu berechnen, wie viel Prozent der neuen Informationen in den Cell State gelangen. Die Input Node wendet hingegen auf dieselben zwei Vektoren eine Tanh-Funktion an, um neue potenzielle Informationen für den Cell State zu erstellen. Wichtig ist, dass die Input Node die potenziellen Informationen für die Speicherung bereitstellt; während das Input Gate bestimmt, zu welchem Anteil die potenziellen Informationen in den Cell State übergehen:[93]

$$\mathbf{i}_t = \sigma\left(\mathbf{W}^{(\mathrm{i})}x_t + \mathbf{U}^{(\mathrm{i})}\mathbf{h}_{t-1} + \mathbf{b}^{(i)}\right) \tag{4.9}$$

$$\tilde{\mathbf{c}}_t = \tanh\left(\mathbf{W}^{(c)}x_t + \mathbf{U}^{(c)}\mathbf{h}_{t-1} + \mathbf{b}^{(c)}\right) \tag{4.10}$$

Nachdem entschieden wurde, was vergessen und was gespeichert werden soll, wird der Cell State wie folgt aktualisiert:[94]

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \tag{4.11}$$

wobei $\odot$ das Hadamard-Produkt ist. Mit diesem Operator wird das elementweise Produkt gebildet. Im Weiteren entscheidet das Output Gate $\mathbf{o}_t$, welche Informationen aus dem Zellzustand ausgegeben werden sollen. Gleichermaßen wie bei $\mathbf{f}_t$ und $\mathbf{i}_t$ werden die neuen Eingaben und der vergangene versteckte Zustand als gewichtete Summe durch die Sigmoidfunktion geleitet:[95]

---

[92] Vgl. Goodfellow et al. (2016), S. 406.
[93] Vgl. Zhang et al. (2023), S. 371.
[94] Vgl. ebenda, S. 372.
[95] Vgl. Géron (2023), S. 749.

$$\mathbf{o}_t = \sigma\left(\mathbf{W}^{(o)}\mathbf{x}_t + \mathbf{U}^{(o)}\mathbf{h}_{t-1} + \mathbf{b}^{(o)}\right) \tag{4.12}$$

Bevor das LSTM-Modul den neuen versteckten Zustand ausgibt, wird abschließend eine Tanh-Funktion auf den Cell State angewendet, um die neuen potenziellen Informationen für den Kurzzeitspeicher auf einen Wert zwischen -1 und 1 zu transformieren. Der neue versteckte Zustand wird demennach wie folgt berechnet:[96]

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{4.13}$$

Wenn das Output Gate nahe 1 liegt, kann der Cell State ungehindert auf die nachfolgenden Schichten wirken. Ist das Output Gate nahe 0, wird verhindert, dass der aktuelle Cell State andere Schichten beeinflusst. Es ist wichtig zu beachten, dass der Cell State über viele Zeitschritte hinweg Informationen akkumulieren kann − ohne das restliche Netzwerk zu beeinflussen. Sobald das Output Gate jedoch von nahe 0 zu nahe 1 wechselt, kann sich der angesammelte Speicherzustand plötzlich auf das Netzwerk auswirken.[97]

Am Ende wird der neue versteckte Zustand zum einen an das LSTM-Modul der nächsten Periode und zum anderen zur Vorhersage der Ausgabeschicht weitergeben. Anschließend wird der Wert der Ausgabeschicht auf die gleiche Weise wie beim RNN berechnet:[98]

$$\hat{\mathbf{y}}_t = g\left(\mathbf{W}^{(2)}\mathbf{h}_t + \mathbf{b}^{(2)}\right) \tag{4.14}$$

Das beste Modell entspricht einem KNN mit optimalen Gewichten. Ein entscheidender Faktor für die Modellqualität ist die Vermeidung von Überanpassung (Overfitting), da LSTMs aufgrund ihrer hohen Kapazität dazu neigen können, sich stark an die Trainingsdaten anzupassen. Um eine gute Generalisierung zu gewährleisten, müssen die historischen Daten in einen Trainingsdatensatz, einen unabhängigen Validierungsdatensatz und einen unabhängigen Testdatensatz unterteilt werden. Da LSTMs speziell für Zeitreihendaten entwickelt wurden, ist eine naive zufällige Aufteilung der Daten nicht sinnvoll. Stattdessen werden häufig spezielle Verfahren wie das rollierende Fensterverfahren (Rolling Windows) verwendet, um sicherzustellen, dass das Modell nicht auf zukünftige Daten zugreift. Eine sorgfältige Wahl der Netzarchitektur, der Regularisierungstechniken und der Hyperparameter ist entscheidend, um ein leistungsfähiges LSTM-Netzwerk zu trainieren, das sowohl vergangene Abhängigkeiten erfasst als auch robuste Vorhersagen trifft.

[96] Vgl. Géron (2023), S. 750.
[97] Vgl. Zhang et al. (2023), S. 372f.
[98] Vgl. Géron (2023), S. 750.

5 Empirische Ergebnisse

5.1 Datenbasis

Der für diese Analyse verwendete Datensatz wurde über Onvista Media von Xetra bezogen. Der Deutsche Aktienindex (DAX) liegt dabei als Performance-Index vor und bildet folglich den bereinigten Wertzuwachs der 40 größten Unternehmen in Deutschland ab. Die Unternehmen werden nach ihrer Streubesitz-Marktkapitalisierung ausgewählt und müssen bestimmte Mindestanforderungen an Qualität und Rentabilität erfüllen. Insbesondere in Bezug auf die Auswahlkriterien sowie die Zusammensetzung gab es bei der Indexberechnung in den vergangenen Jahren einige Anpassungen.[99] Die historischen Indexstände für den DAX-Performance-Index reichen bis zum 30. Dezember 1987 zurück, an dem der DAX mit einem Indexstand von 1.000 Punkten fixiert wurde.[100]

Tab. 1: Deskriptive Statistiken

	DAX-Schlusskurs	Log-Rendite	Historische Volatilität
Anzahl	9.178	9.177	9.147
Durchschnitt	6.686,30	0,03%	19,50%
Standardabweichung	4.352,42	1,38%	10,07%
Minimum	931,18	-13,71%	4,32%
25% Quantil	2.827,51	-0,61%	12,89%
50% Quantil	5.811,62	0,08%	16,75%
75% Quantil	9.889,63	0,73%	22,73%
Maximum	18.492,49	10,80%	75,21%
Schiefe	0,64	-0,28	1,94
Kurtosis	-0,63	6,81	4,85
ADF Statistik	0,44	-16,38*	-6,39*

Hinweis: *, ** und *** geben die Signifikanz auf dem Niveau von 1 %, 5 % bzw. 10 % an.

Quelle: Eigene Darstellung mit Daten der Onvista Media

Im Folgenden werden die täglichen DAX-Schlusskurse in einem Zeitraum von über 36 Jahren (04. Januar 1988 bis 07. Mai 2024) betrachtet. Die DAX-Zeitreihe umfasst somit 9.178 Schlusskurse. Basierend darauf werden gemäß (2.2) die täglichen logarithmischen Renditen berechnet. Diese werden wiederum genutzt, um die tägliche 30-Tage historische Volatilität entsprechend (2.3) zu bestimmen. Folglich liegen für die historischen Volatilitäten 30 Datenpunkte weniger als für die logarithmischen Renditen vor.

[99] Vgl. STOXX (2023), S. 33f.
[100] Vgl. Deutsche Börse (2024), Absatz 2.

Anstelle des Volatilitätsindex VDAX-NEW wird die historische Volatilität berechnet, da GARCH-Modelle auf logarithmischen Differenzen basieren. Verwendet man statt der DAX-Renditen die Differenzen des VDAX-NEW, würde das Modell nicht die tatsächliche Marktvolatilität prognostizieren, sondern vielmehr die Volatilität der Volatilität. Die gewählte Vorgehensweise ermöglicht es, die aus dem Modell abgeleitete prognostizierte Volatilität direkt mit der impliziten Volatilität des VDAX-NEW zu vergleichen und so Rückschlüsse über mögliche Marktineffizienzen abzuleiten.

5.2 Allgemeine Vorgehensweise

Da GARCH-Modelle eine Verallgemeinerung der ARCH-Modelle darstellen und LSTMs aufgrund ihrer Konstruktion besser für Zeitreihenanalysen geeignet sind als FFNs oder konventionelle RNNs, konzentriert sich die empirische Analyse auf den Vergleich zwischen LSTM-Netzwerken und GARCH(p,q)-Modellen. Für den Vergleich werden die LSTM-Modelle ausschließlich auf Basis von täglichen logarithmischen Renditen und täglichen 30-Tage historischen Volatilitäten trainiert. Die Modelle prognostizieren somit die tägliche Volatilität für den jeweils folgenden Tag (Ein-Schritt-Prognose).

Für die ökonometrische Modellierung wird die Stationarität der Finanzmarktzeitreihen vorausgesetzt. Die Nullhypothese des Augmented Dickey-Fuller (ADF)-Tests (Spezifikation: mit Konstante; ohne Trend) kann auf dem 1%-Signifikanzniveau sowohl für die historische Volatilität als auch für die logarithmischen DAX-Renditen abgelehnt werden (siehe Tab. 1). Somit sind beide Zeitreihen stationär.

Für die Modellierung mittels KNN wird der Datensatz im Verhältnis 50:25:25 in einen Trainings-, Validierungs- und Testdatensatz unterteilt. Der Trainingsdatensatz umfasst den Zeitraum vom 16. Februar 1988 bis zum 27. April 2006 und wird für den Trainingsprozess verwendet. Das Netzwerk lernt aus diesen Daten, indem es seine Gewichte und Bias anpasst und die Verlustfunktion minimiert. Der Validierungsdatensatz erstreckt sich vom 28. April 2006 bis zum 30. April 2015 und wird verwendet, um die Leistung des Netzwerks während des Trainings zu überwachen. Folglich wird dieser Datensatz für das Hyperparameter-Tuning verwendet. Der Testdatensatz deckt den Zeitraum vom 4. Mai 2015 bis zum 7. Mai 2024 ab und dient zur endgültigen Bewertung der Netzeinstellung nach dem Abschluss des Trainings. Die Testdaten hat das Netzwerk während des Trainings nicht gesehen, weshalb eine unvoreingenommene Abschätzung der tatsächlichen Leistungsfähigkeit des Netzwerks möglich ist und die Generalisierungsfähigkeit adäquat bewertet werden kann.

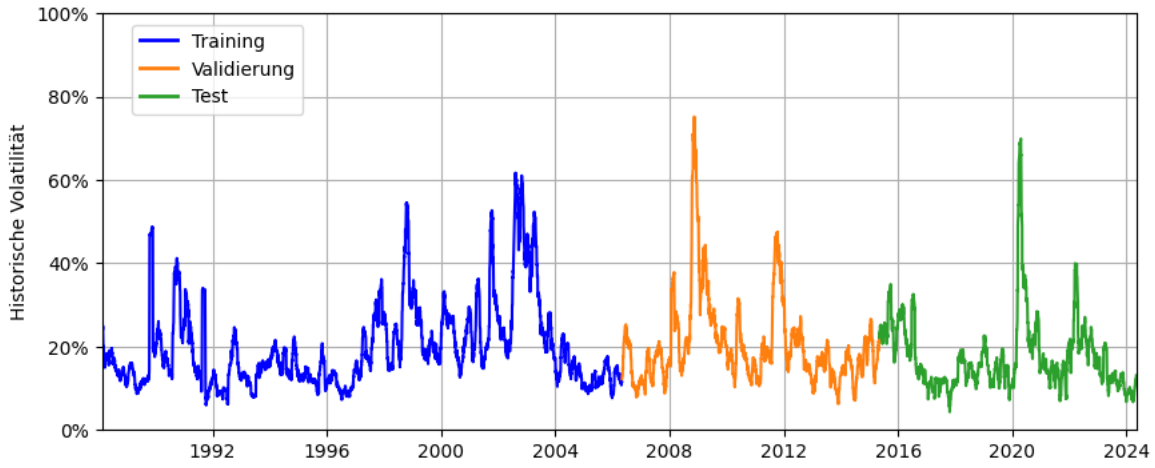

Abb. 14: Historische 30-Tage Volatilität (Trainings-, Validierungs- & Testdatensatz)
Quelle: Eigene Darstellung mit Daten der Onvista Media

Die gewählte Aufteilung des Datensatzes ermöglicht einerseits die Validierung des LSTM-Netzes an einem Datensatz, der in seiner Größe dem Testdatensatz entspricht, und stellt andererseits sicher, dass mindestens ein extremes Ereignis in jedem Datensatz enthalten ist. Der Trainingsdatensatz umfasst neben der deutschen Wiedervereinigung vor allem die Asienkrise um den Beinahe-Kollaps des Long-Term Capital Management Hedgefonds sowie die Terroranschläge vom 11. September 2001. Der Validierungssatz berücksichtigt insbesondere die Auswirkungen der globalen Finanzkrise 2007/08 sowie die der europäischen Staatsschuldenkrise 2010-12. Der Testdatensatz umfasst neben der Reaktion auf das Brexit-Votum vor allem die Auswirkungen der Lockdowns während der COVID-19-Pandemie sowie den Ukraine-Krieg.[101]

Die Prognosegüte der historischen Volatilität des DAX wird sowohl innerhalb als auch außerhalb der Stichprobe anhand von zwei Fehlermaßen bewertet: dem mittleren quadratischen Fehler (Root Mean Squared Error, RMSE) und dem mittleren absoluten Fehler (Mean Absolute Error, MAE). Diese Maße werden wie folgt ermittelt:[102]

$$RMSE = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(\sigma_t - \hat{\sigma}_t)^2} \qquad (5.1)$$

$$MAE = \frac{1}{T}\sum_{t=1}^{T}|\sigma_t - \hat{\sigma}_t|$$

[101] Vgl. DWS (2024), Absatz 5.
[102] In Anlehnung an Greene (2019), S. 128.

Während der MAE den durchschnittlichen Fehler misst und alle Fehler gleich gewichtet, bewertet der RMSE größere Fehler stärker. Da hohe Abweichungen bei der Volatilität besonders unerwünscht sind, wird der MSE zu Optimierung des LSTMs herangezogen. Der MSE entspricht dem quadrierten RMSE in (4.3) und wird bevorzugt, da er große Fehler stärker gewichtet, jedoch aufgrund des Verzichts auf die Quadratwurzel sparsamer zu berechnen ist.[103]

5.3 Vorgehen bei den GARCH-Modellen

Die Schätzung der GARCH-Modelle sowie deren asymmetrischen Erweiterungen erfolgte mit Python (Version 3.10.12). Für deren Optimierung wurde der Standard-Optimierungsalgorithmus von scipy in der arch-Bibliothek durch den SLSQP-Algorithmus ersetzt.[104]

Für die Durchführung der GARCH-Modellierung wurde zunächst überprüft, ob die erforderlichen Annahmen erfüllt sind. Die Stationarität der logarithmischen Renditen des DAX kann mittels des ADF-Tests nicht angelehnt werden. Zudem zeigt Abb. 5, dass die Volatilität tendenziell zum Mittelwert zurückkehrt. Die Autokorrelationsfunktion in Abb. 4 und der Durbin-Watson-Test mit einem Wert von etwa 2 für die logarithmischen Renditen belegen, dass keine Autokorrelation erster oder höherer Ordnung vorliegt. Darüber hinaus weist die Zeitreihe ein endliches viertes Moment auf, was auf eine endliche Varianz und eine stabile Modellparametrisierung hinweist.

Als nächstes wurde die optimale Modellordnung für die GARCH(p,q)-Modelle mittels dem Akaike-Informationskriterium (AIC) und dem Bayes'schen Informationskriterium (BIC) bestimmt. Diese Kriterien wurden für verschiedene Kombinationen von p und q berechnet, wobei allen Modellen eine Normalverteilung zugrundeliegt. Die AIC- und BIC-Werte zeigen, dass ein Modell mit $p = 2$ und $q = 1$ optimal ist (siehe Abb. 15). Diese Modellordnung wurden als Grundlage für die weitere Analyse des klassischen GARCH-Modells gewählt.

Darüber hinaus sollen asymmetrische GARCH-Modelle geschätzt werden, um einen Vergleich zwischen den GARCH(p, q)-Modellen sowie deren Erweiterungen zu ermöglichen. Neben dem GARCH(2,1) mit Normalverteilung wird ein GARCH(2,1) mit einer t-Verteilung geschätzt. Als Erweiterungen wurden das symmetrische und asymmetrische EGARCH sowie das asymmetrische GJR-GARCH gewählt. Während die symmetrischen EGARCH-Modelle sowohl unter der Normal- als auch t-Verteilungsannahme geschätzt werden, werden die asymmetrischen GARCH-Erweiterungen nur unter der Annahme einer Normalverteilung geschätzt.

[103] Vgl. Géron (2023), S. 77f.
[104] Vgl. Sheppard (2021), bashtage/arch: Release 4.18 (Version v4.18).

Dies liegt in der Tatsache begründet, dass für asymmetrische EGARCH und GJR-GARCH mit t-Verteilungsannahme keine sinnvollen Ergebnisse berechnet wurden, da der SLSQP-Algorithmus keine optimale Lösung finden konnte. Die beiden Informationskriterien AIC und BIC zeigen, dass ein EGARCH- bzw. GJR-GARCH-Modell mit $p = 1$ und $q = 1$ optimal ist (siehe Abb. 16).

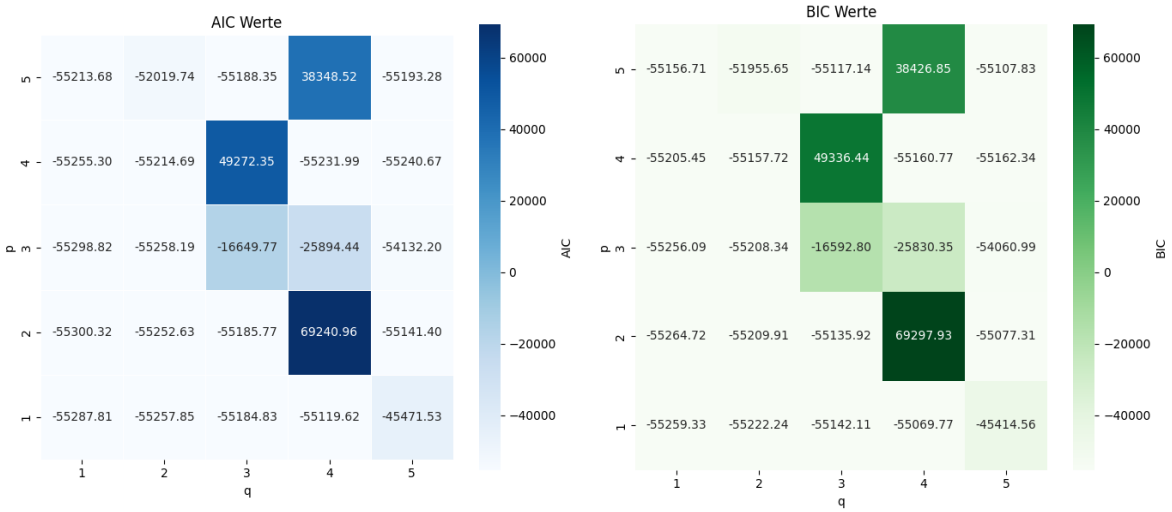

Abb. 15: AIC- und BIC-Werte für das GARCH(p,q)-Modell mit Normalverteilung
Quelle: Eigene Darstellung mit Daten der Onvista Media

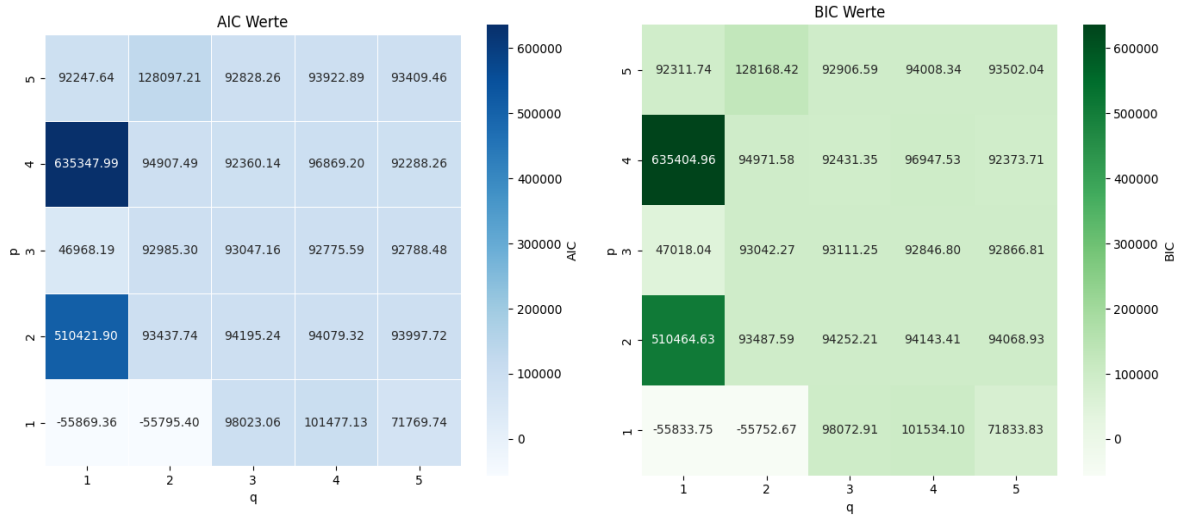

Abb. 16: AIC- und BIC-Werte für das EGARCH(p,q)-Modell mit t-Verteilung
Quelle: Eigene Darstellung mit Daten der Onvista Media

5.4 Vorgehen beim LSTM-Netzwerk

Die Umsetzung des LSTM-Netzwerks in Python erfolgte unter Verwendung der Bibliotheken TensorFlow und Keras. Insbesondere Keras ermöglicht eine benutzerfreundliche und flexible Implementierung von Deep-Learning-Verfahren − einschließlich LSTM. Zusätzlich wurde Scikit-learn für die Datenvorverarbeitung und die Berechnung der Fehlermaße verwendet.

Ein wesentlicher Schritt im Aufbau eines leistungsfähigen neuronalen Netzes ist die Feinabstimmung der Hyperparameter. Hyperparameter sind Parameter, die festgelegt werden müssen und den Lernprozess des Netzwerks maßgeblich beeinflussen. Während es gängige Praxis ist, alle möglichen Parameterkombinationen zu evaluieren und diejenige zu wählen, die auf dem Validierungsdatensatz am besten abschneidet; ist dieses Verfahren bei einer großen Anzahl von Parametern oft rechnerisch sehr aufwendig. Aus diesem Grund wurde in diesem Beitrag ein initiales Architekturdesign festgelegt, um dann die optimalen Werte für eine reduzierte Menge an Hyperparametern zu bestimmen.[105]

Zunächst wurde ein LSTM-Netzwerk in Keras implementiert, dass eine einzige versteckte Schicht aufweist. Bezüglich der Aktivierungsfunktionen wurden eine Tanh-Funktion in der versteckten Schicht und eine lineare Funktion in der Ausgabeschicht verwendet. Als Optimierer kam Adam (Adaptive Moment Estimation) zum Einsatz, eine Variante des Mini-Batch-Gradient-Descents, die die Lernrate in jeder Iteration für jeden Modellparameter anpasst. Zu den zu optimierenden Hyperparametern gehören die Anzahl der Merkmale, die Anzahl der versteckten Neuronen, die Batchgröße und die Anzahl der Epochen. Diese Hyperparameter wurden gewählt, da sie einen erheblichen Einfluss auf die Leistung des LSTM-Netzwerks ausüben.[106]

Um die Suche nach den optimalen Parameterwerten zu beschleunigen, wurde als Erstes die beste Kombination aus der Anzahl der Merkmale und der versteckten Neuronen ermittelt, da diese Parameter das Lernverhalten des Netzwerks am meisten beeinflussen. Da das LSTM für die Vorhersage der Volatilität mit den logarithmischen Renditen und der historischen 30-Tage Volatilität trainiert werden soll, steht die Anzahl der Merkmale bereits fest. Somit übersetzt sich die Wahl der optimalen Merkmalszahl, ähnlich wie bei den GARCH-Modellen, in die Anzahl der pro Merkmal einzuschließenden Lags. Um die Komplexität zu reduzieren, wurde angenommen, dass diese Anzahl für beide Merkmale gleich hoch ist. Das Modell wurde anschließend für jede Kombination aus Anzahl der Neuronen und Lags pro Merkmal validiert.

[105] Vgl. Géron (2023), S. 64f.
[106] Vgl. Zhang et al. (2023), S. 532ff.

Dabei wurde ein Seed gesetzt, um sicherzustellen, dass die Unterschiede zwischen den Ergebnissen nicht auf Zufälligkeit zurückzuführen und die Ergebnisse reproduzierbar sind. Die Batchgröße wurde zunächst auf 32 gesetzt und das Modell wurde für maximal 1.000 Epochen trainiert, wobei ein vorzeitiger Abbruch (Early Stop) erfolgte, falls der Validierungsfehler für 100 aufeinanderfolgende Epochen nicht abnahm. In diesem Fall wurden die besten Gewichte übernommen und der minimale Fehler ausgegeben.

Tab. 2: RMSE für verschiedene Kombinationen von Neuronen und Lags pro Merkmal

Anzahl der versteckten Neuronen	Lags pro Merkmal					
	1	2	3	4	5	6
2	0,01018747	0,00147062	0,00435958	0,0049861	0,00628513	0,00731519
4	0,00995417	0,00057141	0,00086963	0,00210653	0,00368276	0,00472432
6	0,00981412	0,0003479	0,00108636	0,00159447	0,00243002	0,00437942
8	0,00976349	0,00036379	0,00092743	0,00128783	0,0050566	0,00661789
10	0,00972099	0,00024328	0,00117736	0,00106795	0,00503403	0,00539862
12	0,00974186	0,00026348	0,00087261	0,00075776	0,00149915	0,00574874
14	0,00985338	0,00019024	0,00074554	0,00087721	0,00160332	0,00141042
16	0,00971183	0,00018089	0,00056598	0,0010935	0,00182787	0,00158066
18	0,00974868	0,00038783	0,00040178	0,00116503	0,00117681	0,00558518
20	0,00973082	0,00029977	0,00121201	0,00089139	0,00099174	0,00576099

Quelle: Eigene Darstellung mit Daten der Onvista Media

Die Validierungsergebnisse − gemessen mittels RMSE − sind in Tab. 2 dargestellt. Hierbei zeigt sich, dass eine höhere Anzahl an versteckten Neuronen und Lags pro Merkmal die Wahrscheinlichkeit erhöht, dass das Modell die Trainingsdaten überanpasst. Aus diesem Grund wurde sowohl der RMSE als auch die Modellkomplexität berücksichtigt, um die optimale Kombination auszuwählen. Letztlich wurden 2 Zeitverzögerungen pro Merkmal und 16 versteckte Neuronen gewählt.

Nach der Aktualisierung der anfänglichen Netzarchitektur mit diesen Ergebnissen wurde die optimale Batchgröße bestimmt. Üblicherweise werden hier Potenzen von 2 als Batchgröße verwendet, da diese zu höherer rechnerischer Effizienz führen. Zusätzlich zu diesen Werten wurde auch der Fall betrachtet, in dem die Batchgröße 1 beträgt, was der Verwendung von SGD entspricht, und der Fall, in dem die Batchgröße der Gesamtanzahl der Daten im Trainingsdatensatz (4.572) entspricht − was das Gradientenabstiegsverfahren impliziert. Die optimale Batchgröße wurde auf 128 festgelegt. Obwohl der RMSE für diese Batchgröße nicht der niedrigste ist, hat sich gezeigt, dass sowohl die Prognosegüte als auch die Stabilität des Netzwerks damit optimal sind.[107]

[107] Vgl. Géron (2023), S. 212f.

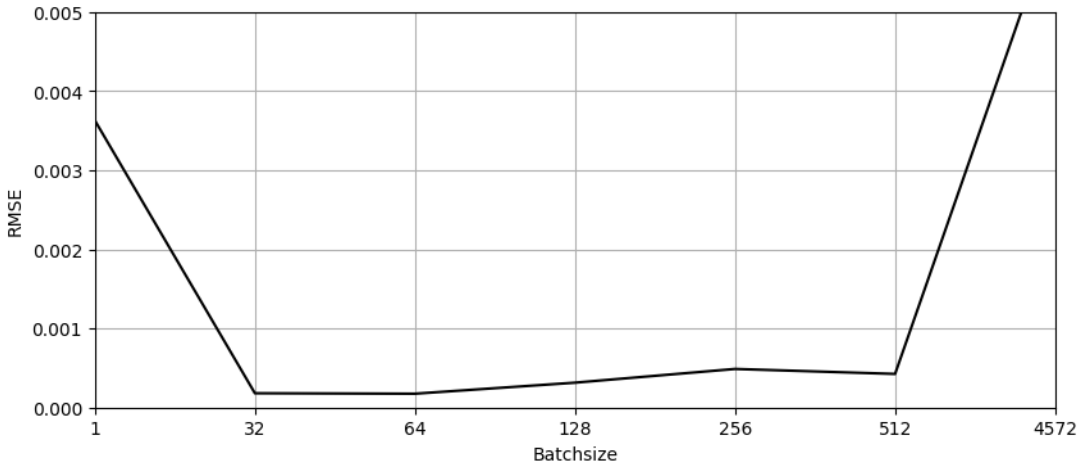

Abb. 17: RMSE für verschiedene Batchgrößen
Quelle: Eigene Darstellung mit Daten der Onvista Media

Im Weiteren wurde die Anzahl der Epochen optimiert. Ziel war es, die Anzahl der Epochen zu bestimmen, nach der das LSTM das beste Out-of-sample-Verhalten zeigt − basierend auf dem Trainings- und Validierungsfehler. In Abb. 18 befinden sich die Verlustkurven mit der abschließenden Wahl der Hyperparameter. Ab etwa 1.000 Epochen stabilisiert sich sowohl der Trainings- als auch der Validierungsfehler und die Differenz zwischen den beiden bleibt relativ gering. Dies deutet darauf hin, dass das Modell eine gute Generalisierungsfähigkeit auf neue Daten besitzt und nicht überanpasst, wenn die Anzahl der Epochen erhöht wird. Daher wird 1.000 als optimaler Wert für diesen letzten Hyperparameter gewählt.

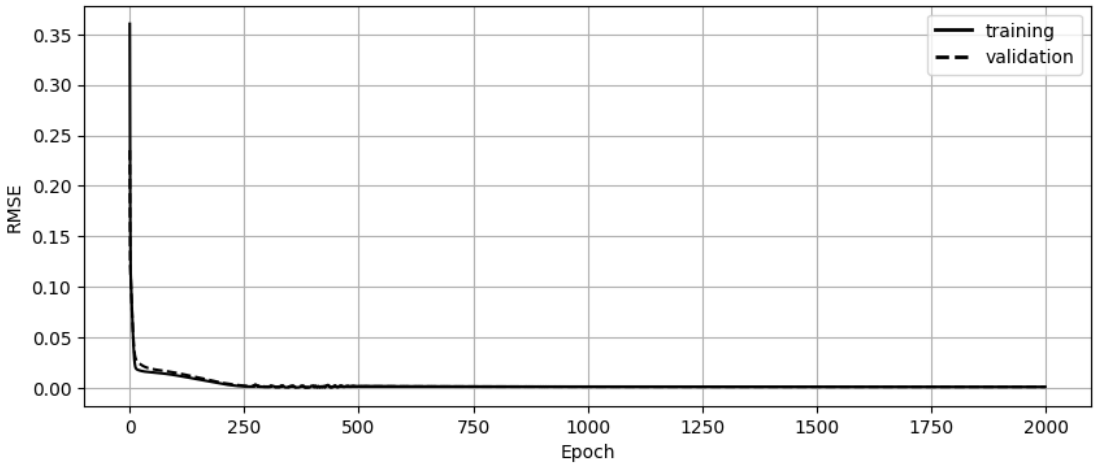

Abb. 18: RMSE-Verlustkurve für Trainings- und Validierungsdatensatz
Quelle: Eigene Darstellung mit Daten der Onvista Media

Abschließend kann die finale Struktur des LSTM-Netzwerks wie folgt beschrieben werden: Das Netzwerk besteht aus einer einzigen versteckten LSTM-Schicht mit 16 Neuronen, gefolgt von einer Ausgabeschicht mit einem Neuron. Zu den finalen Hyperparametern gehören

die Batchgröße von 128, 2 Lags pro Parameter und eine maximale Anzahl von 1.000 Epochen. Die Anzahl der Parameter des Netzwerks ergibt sich durch die Berechnung der Verbindungen innerhalb der LSTM-Schicht sowie zu und von der Ausgabeschicht. Die LSTM-Schicht hat insgesamt 1.216 Parameter, die sich aus den Gewichtsmatrizen und Bias für die Eingabe-, Ausgabe- und versteckte Schicht sowie den Speicherzuständen ergibt. Die Ausgabeschicht fügt weitere 17 Parameter hinzu, die ebenfalls aus den Gewichtungen und dem Bias des einzelnen Neurons entstammen. Insgesamt umfasst das Netzwerk somit 1.233 trainierbare Parameter. Diese Konfiguration wurde gewählt, um eine optimale Balance zwischen Modellkomplexität und Generalisierungsfähigkeit zu gewährleisten.

5.5 Prognoseergebnisse mit GARCH-Modellen

Nach dem Aufbau des LSTM-Netzwerks können nun die Prognoseergebnisse untersucht werden. Da die Prognosen außerhalb der Stichprobe auf Gewichten und Parametern basieren, die aus den Daten des Trainings- und Validierungssatzes gelernt wurden, werden diese beiden Datensätze im Folgenden zusammengefasst und als Trainingsdatensatz bezeichnet.

Die Schätzung des GARCH(2,1)-Modells mit Normalverteilung zeigt, dass die Volatilitätsparameter $\hat{\alpha}_0$, $\hat{\alpha}_1$, $\hat{\alpha}_2$ und $\hat{\beta}_1$ auf dem 1%-Niveau signifikant sind. Ferner ist zu sehen, dass die aktuelle Volatilität stark von den vergangenen Volatilitäten ($\hat{\beta}_1 = 0{,}88$) und nur schwach von den vergangenen Schocks abhängt ($\hat{\alpha}_1, \hat{\alpha}_2 = 0{,}05$). Unter Anwendung von (3.5) beträgt die unbedingte Varianz dieses GARCH(2,1)-Modells 22,45 %. Da im EGARCH(1,1,1)-Modell der Parameter $\hat{\gamma} < 0$ und im GJR-GARCH(1,1,1)-Modell $\hat{\gamma} > 0$ ist, beeinflussen positive Kursänderungen die Volatilität weniger stark als negative Schocks. Damit wurde der Leverage-Effekt am deutschen Aktienmarkt festgestellt.

Tab. 3: Geschätzte Parameter für verschiedene GARCH-Modelle

Modell	Verteilung	$\hat{\alpha}_0$	$\hat{\alpha}_1$	$\hat{\alpha}_2$	$\hat{\beta}_1$	$\hat{\gamma}$
GARCH(2,1)	**Normal**	0,000004*	0,050000*	0,050000*	0,880000*	-
GARCH(2,1)	**t**	0,000002*	0,036312*	0,065856*	0,888943*	-
EGARCH(1,1)	**Normal**	-0,195454*	0,185225*	-	0,976964*	-
EGARCH(1,1)	**t**	-0,099099*	0,171184*	-	0,988080*	-
EGARCH(1,1,1)	**Normal**	-0,199181*	0,127680*	-	0,977035*	-0,092277*
GJR-GARCH(1,1,1)	**Normal**	0,001904*	0,000000*	-	0,000000*	2,000000*

Hinweis: *, ** und *** geben die Signifikanz auf dem Niveau von 1 %, 5 % bzw. 10 % an.

Quelle: Eigene Darstellung mit Daten der Onvista Media

Die angepassten und vorhergesagten historischen Volatilitäten des Modells sind in Abb. 19 dargestellt. Dabei wird sichtbar, dass das Modell die Volatilität sowohl in Zeiten hoher als auch in Zeiten niedriger Volatilität tendenziell überschätzt. Dies wird ferner mit Blick auf die Fehlermaße (siehe Tab. 4) deutlich, da der RMSE sowie der MAE sowohl In- als auch Out-of-sample weit von null entfernt sind.

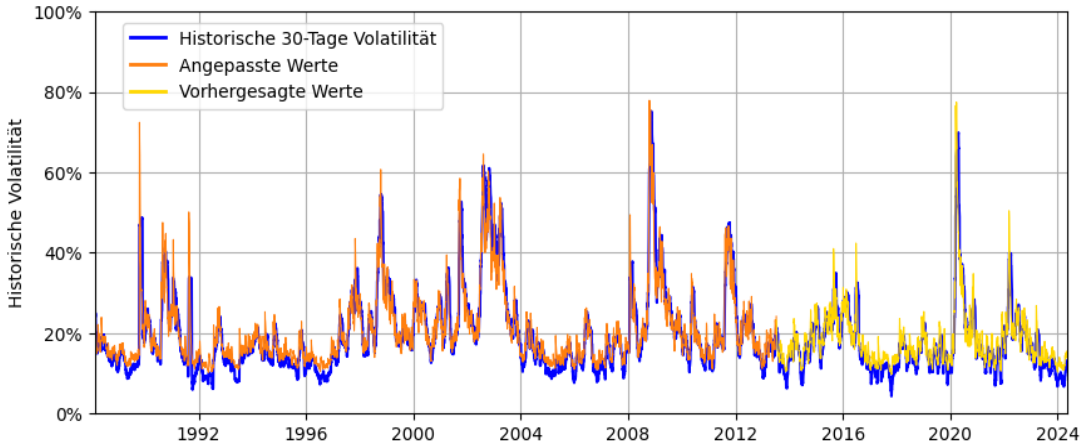

Abb. 19: Angepasste und vorhergesagte GARCH(2,1)-Werte (Normalverteilung)
Quelle: Eigene Darstellung mit Daten der Onvista Media

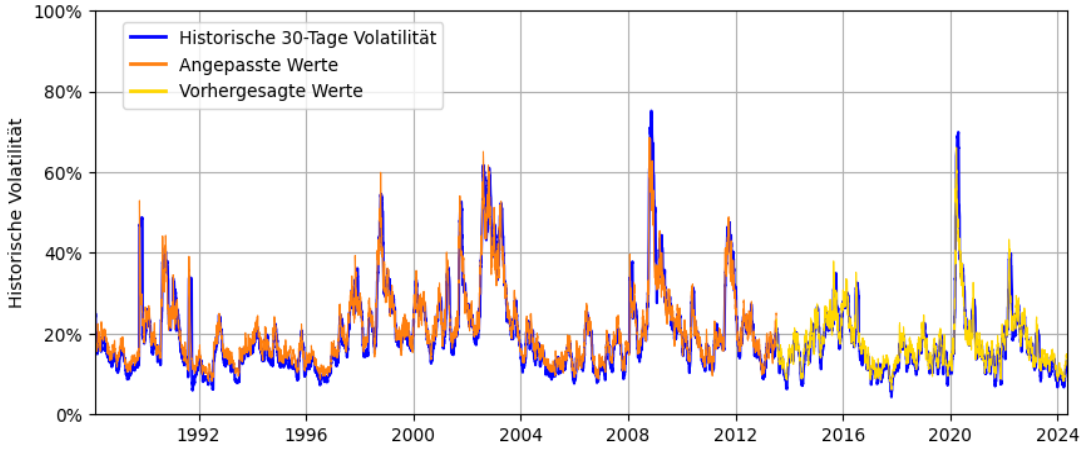

Abb. 20: Angepasste und vorhergesagte EGARCH(1,1)-Werte (t-Verteilung)
Quelle: Eigene Darstellung mit Daten der Onvista Media

Tab. 4 enthält die Prognosegüte der einzelnen GARCH-Modelle. Es wird deutlich, dass das symmetrische EGARCH(1,1) mit der t-Verteilung die geringsten Ein-Schritt-Prognosefehler sowohl In-sample als auch Out-of-sample im Vergleich zu den anderen GARCH-Modellen liefert. Dies zeigt sich auch in Abb. 20, in dem die angepassten und vorhergesagten Volatilitäten deutlich näher bei den empirischen Werten liegen.

Tab. 4: In-sample und Out-of sample Fehlermaße für verschiedene GARCH-Modelle

Modell	Verteilung	RMSE		MAE	
		In-sample	Out-of-sample	In-sample	Out-of-sample
GARCH(2,1)	Normal	0,037379	0,035860	0,026229	0,026092
GARCH(2,1)	t	0,034252	0,033126	0,023446	0,023397
EGARCH(1,1)	Normal	0,040255	0,035511	0,027695	0,024596
EGARCH(1,1)	t	**0,033326**	**0,029840**	**0,022946**	**0,020471**
EGARCH(1,1,1)	Normal	0,048361	0,043670	0,033255	0,030260
GJR-GARCH(1,1,1	Normal	0,046873	0,046174	0,031896	0,031807

Quelle: Eigene Darstellung mit Daten der Onvista Media

5.6 Prognoseergebnisse mit LSTM-Netzwerken

Das LSTM wurde mit den in Kapitel 5.4 beschriebenen Hyperparametern auf Basis des Trainingsdatensatzes trainiert. Da die neuronalen Netze keinen parametrischen Ansatz verfolgen, müssen an dieser Stelle keine weiteren Annahmen geprüft und keine Parameter auf Signifikanz getestet werden. Die Leistungsfähigkeit des LSTM zeigt sich insbesondere in den niedrigen Fehlern bei der Ein-Schritt-Prognose (siehe Tab. 5). Sowohl auf den Trainingsdaten als auch auf den Testdaten erreicht das neuronale Netz eine bemerkenswert hohe Prognosegüte. Dies zeigt sich vor allem darin, dass die angepassten und vorhergesagten Werte in Abb. 21 nahezu exakt der historischen 30-Tage DAX-Volatilität entsprechen. Dies weist auf eine exzellente Modellanpassung hin − sowohl In-sample als auch Out-of-sample.

Tab. 5: In-sample und Out-of-sample Fehlermaße des LSTMs

Index	RMSE		MAE	
	In-sample	Out-of-sample	In-sample	Out-of-sample
DAX	0,000273	0,000225	0,000163	0,000133

Quelle: Eigene Darstellung mit Daten der Onvista Media

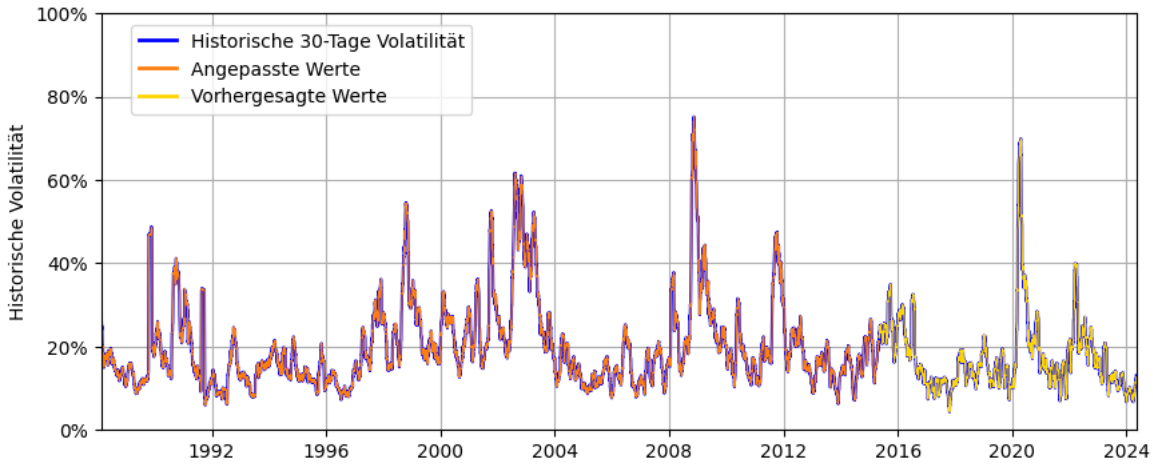

Abb. 21: Angepasste und vorhergesagte LSTM-Werte
Quelle: Eigene Darstellung mit Daten der Onvista Media

Um die Stabilität und Generalisierbarkeit des LSTMs weiter zu überprüfen, wurde das Netzwerk mit denselben Hyperparametern auf weitere Aktienindizes bedeutender internationaler Länder wie z. B. dem S&P500, Nikkei 225 und Hang Seng angewendet. Hierbei wurden die verschiedenen Zeitreihen der Indizes erneut im Verhältnis 50:25:25 in Trainings-, Validierungs- und Testdatensatz unterteilt. Der Trainingsdatensatz erstreckt sich vom 13. Februar 1998 bis zum 23. März 2011. Der Validierungsdatensatz umfasst den Zeitraum vom 24. März 2011 bis zum 13. Oktober 2017. Der Testdatensatz deckt den Zeitraum vom 16. Oktober 2017 bis zum 7. Mai 2024 ab. Folglich werden im Trainingsdatensatz erneut Aktienmarktschocks wie die Terroranschläge vom 11. September 2001 und die Weltfinanzkrise sowie im Testdatensatz u. a. die Lockdowns der COVID-19-Pandemie und der Ukraine-Krieg berücksichtigt.

Tab. 6: In-sample und Out-of-sample Fehlermaße des LSTMs für internationale Aktienindizes

Index	RMSE		MAE	
	In-sample	Out-of-sample	In-sample	Out-of-sample
ATX	0,001320	0,001306	0,000809	0,000739
CAC	0,000219	0,000229	0,000110	0,000095
FTSE MIB	0,000152	0,000393	0,000080	0,000088
HSI	0,001247	0,000982	0,000631	0,000549
IBEX	0,000385	0,000439	0,000254	0,000186
NKY	0,000195	0,000068	0,000073	0,000045
SMI	0,000448	0,000388	0,000335	0,000314
SPX	0,000166	0,000216	0,000108	0,000121
SX5E	0,000487	0,000566	0,000345	0,000305
UKX	0,000076	0,000098	0,000041	0,000041

Quelle: Eigene Darstellung mit Daten der Onvista Media

Die Ergebnisse dieser Untersuchungen waren ebenfalls positiv und konsistent mit den ursprünglichen Ergebnissen, was auf eine hohe Stabilität und Robustheit der Methodik hinweist. Tab. 6 zeigt erneut eine sehr hohe Prognosegüte. Wie in Abb. 21 entsprechen die angepassten und vorhergesagten Werte in Abb. 22 nahezu exakt der historischen Volatilität. Dies impliziert, dass die gewählten Hyperparameter gut generalisieren und das Modell in der Lage ist, auf verschiedenen Datensätzen ähnlich präzise Vorhersagen zu liefern. Die Hyperparameter eignen sich folglich nicht nur für die Volatilitätsprognose des DAX, sondern für Volatilitätsprognosen im Allgemeinen.

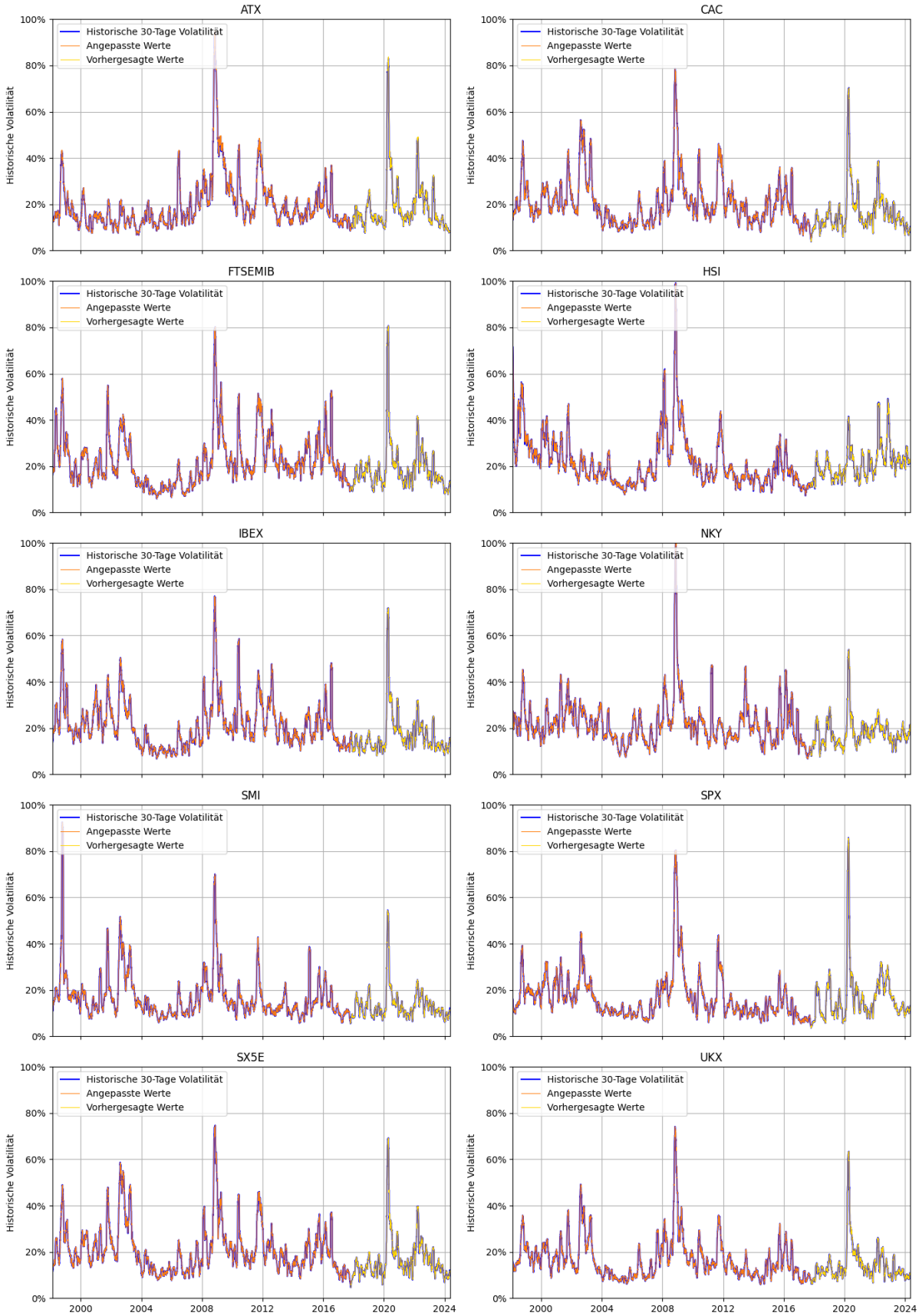

Abb. 22: Angepasste und vorhergesagte LSTM-Werte für internationale Aktienindizes
Quelle: Eigene Darstellung mit Daten der Onvista Media

6 Kritische Analyse und Handlungsempfehlungen

6.1 Vergleich der Prognoseergebnisse von GARCH und LSTM

Die Prognoseergebnisse von GARCH und LSTM unterscheiden sich in ihrer Güte deutlich. Während zeitreihenökonometrische Modelle wie GARCH(2,1) und das EGARCH(1,1) mit einer t-Verteilung signifikante Parameter und einige brauchbare Prognosen liefern; neigen sie dazu, die Volatilität in Zeiten sowohl hoher als auch niedriger Volatilität zu überschätzen. Auch für den deutschen Aktienmarkt gewichten EGARCH-Modelle den Effekt großer Schocks auf die Volatilität tendenziell über. Diese Tendenz wird durch die beiden Fehlermaße RMSE und MAE in Tab. 4 sichtbar, da sowohl die In-sample als auch die Out-of-sample Ein-Schritt-Prognosefehler relativ weit von Null entfernt sind.

Im Vergleich dazu liefert das Künstliche Neuronale Netzwerk sehr genaue Prognosen. Die berechneten Fehlermaße zeigen, dass das LSTM sowohl in den Trainings- als auch in den Testdaten nahezu exakte Vorhersagen der historischen 30-Tage Volatilität ermöglicht (siehe Tab. 5 und Tab. 6). Insbesondere ist die geringere Differenz zwischen den In-sample- und den Out-of-sample-Messungen beim LSTM hervorzuheben. Während die durchschnittlichen Abweichungen für die GARCH-Modelle bei 0,002711 Prozentpunkten für den RMSE und 0,001474 Prozentpunkten für den MAE liegen, betragen sie beim LSTM lediglich 0,000048 bzw. 0,00003 Prozentpunkte. Dies deutet darauf hin, dass neuronale Netze deutlich besser mit Überanpassungen (Overfitting) umgehen können als der Durchschnitt aus den verschiedenen symmetrischen und asymmetrischen GARCH-Modellen.

Ein weiterer Vorteil des LSTM-Netzwerks ist seine Generalisierbarkeit. Das Long Short-Term Memory wurde hierfür auf verschiedene internationale Aktienindizes angewandt und die Prognoseergebnisse waren durchweg konsistent und genauer als bei den ökonometrischen Volatilitätsmodellen. Dies weist auf die hohe Stabilität und Robustheit des LSTMs hin. Im Gegensatz dazu zeigen die GARCH-Modelle eine größere Variabilität in den Vorhersagen und sind weniger adaptiv gegenüber unterschiedlichen Marktbedingungen.

Insgesamt zeigt sich, dass das LSTM-Netzwerk den GARCH-Modellen in Bezug auf die Prognosegenauigkeit und die Handhabung von Überanpassungen überlegen ist. Die Fehlermaße der LSTMs betragen nur etwa 1% derjenigen der GARCH-Modelle und zeigen eine bemerkenswert geringe Differenz zwischen In-sample- und Out-of-sample-Ergebnissen. Diese Vorteile machen das LSTM zu einer sinnvollen Alternative für die Volatilitätsprognose.

6.2 Praktische Relevanz genauerer Volatilitätsprognosen

Die empirische Überlegenheit des LSTM für die Volatilitätsprognose bietet einen erheblichen strategischen Vorteil im Bereich des Optionshandels respektive der dynamischen Absicherung. Eine von Optionshändlern häufig verwendete Strategie für das Risikomanagement ihrer Portfolien ist der sogenannte Delta-Hedge. Dabei wird das Marktpreisrisiko, das durch Schwankungen des zugrundeliegenden Basiswerts entsteht, minimiert, indem die Delta-Position des Portfolios kontinuierlich ausgeglichen wird. Das Delta beschreibt dabei die Sensitivität des Portfoliowerts gegenüber Preisänderungen des Basiswerts. Ein deltaneutrales Portfolio reagiert folglich nicht mehr auf die Preisbewegungen des Basiswerts. Stattdessen ist das Portfolio den Risiken ausgesetzt, die durch die anderen Greeks[108] entstehen. Dem Händler geht es aber insbesondere um die im Optionspreis enthaltene implizite Volatilität, die entweder zu hoch oder zu niedrig ist. Demzufolge sind Optionen nur dann zu verkaufen, wenn zukünftig mit einer geringeren Volatilität gerechnet wird, und umgekehrt.[109]

In der Praxis hängt die Effektivität der Delta-Hedge-Strategie maßgeblich von der Genauigkeit der Volatilitätsprognose ab. Je präziser die Prognose, desto genauer können Händler das Delta bestimmen und dementsprechend die erforderlichen Hedge-Anpassungen vornehmen. Traditionelle GARCH-Modelle würden an dieser Stelle falsche Anreize setzen, da sie die Volatilität tendenziell überschätzen. Durch die präzisere LSTM-Schätzung können Händler effektivere Delta-Hedges implementieren.

Darüber hinaus könnte die verbesserte Genauigkeit der LSTM-Prognose Auswirkungen auf das Volatility-Risk-Premium (VRP) haben. Die Marktteilnehmer verlangen beim Verkauf von Optionen eine zusätzliche, relativ hohe Prämie für die Unsicherheit und das Risiko zukünftiger Volatilität. Daraus folgt, dass Optionspreise tendenziell zu teuer sind, weil Angst besteht, die zukünftige Volatilität falsch zu bemessen. Durch eine akkuratere Volatilitätsprognose, wie sie Künstliche Neuronale Netze wie das LSTM ermöglichen, könnte das VRP reduziert und Optionen marktgerechter bepreist werden. Wenn Marktteilnehmer genauer die zukünftige Volatilität abschätzen könnten, würde dies zu einer geringeren Unsicherheit führen und damit die Prämie, die sie für das Halten von Optionen verlangen, senken. Diese Reduzierung könnte dazu beitragen, dass die Optionspreise weniger durch das VRP verzerrt werden. Folglich würden auch die Kosten für den Delta-Hedge fallen.

---

[108] Die Greeks sind die partiellen Ableitungen des Optionspreises nach den Modellparametern im Black-Scholes-Modell, die die Sensitivitäten der Option gegenüber diesen Parametern quantifizieren.
[109] Vgl. Boyle / McDougall (2019), S. 111ff.

6.3 Herausforderungen beim Einsatz von Deep Learning im Bankensektor

Wenngleich die aufgeführten Chancen für eine Anwendung von LSTMs in Banken sprechen, müssen gleichermaßen die Grenzen der KNN aufgezeigt werden. Ein oft diskutiertes Problem neuronaler Netze ist ihre „Black Box"-Natur. Gemeint ist damit die Unklarheit, wie oder warum ein Netzwerk zu einem bestimmten Ergebnis gelangt. Zwar kann wie in Kapitel 4.4 gezeigt wurde, die Berechnung der LSTM-Prognose einfach nachvollzogen werden, dennoch ist es schwierig zu erklären, warum ein bestimmter Prognosewert durch das Netz ausgegeben wird. Wenn das LSTM eine Volatilitätsspitze prognostiziert, ist es schwierig nachzuvollziehen, welche spezifischen Muster oder Datenpunkte zu dieser Vorhersage geführt haben. Im Vergleich dazu sind die ökonometrischen Modelle viel transparenter und ermöglichen anhand der geschätzten Modellparameter eine klare Interpretation der Ergebnisse.[110] Diese Intransparenz stellt ein erhebliches Hindernis für Kreditinstitute dar, die Deep-Learning-Modelle zur Volatilitätsprognose einsetzen möchten. Gemäß 4.3.5. der Mindestanforderungen an das Risikomanagement (MaRisk) müssen Banken in der Lage sein, die Gründe für bestimmte Prognosen bzw. Modellergebnisse klar darzulegen. Folglich nützt einer Bank aus regulatorischer Sicht das beste Modell wenig, wenn sie es nicht hinreichend erklären kann.[111]

Überdies muss die Entwicklungszeit für das Netzwerk betrachtet werden. Obwohl es Bibliotheken wie Keras gibt, die die Entwicklung von KNN relativ einfach gestalten, kann es für eine Bank Situationen geben, in denen mehr Kontrolle über die Detaileinstellungen des Netzwerks erforderlich ist. Dies ist insbesondere der Fall, wenn komplexere Architekturdesigns oder individuelle Hyperparameter verwendet werden sollen. In solchen Fällen müsste auf Tensorflow zurückgegriffen werden, das zwar mehr Möglichkeiten bietet, aber auch komplexer ist und eine längere Entwicklungszeit erfordert.[112]

Für Banken, die beispielsweise LSTM-Netzwerke zu Volatilitätsprognose einsetzen möchten, stellt sich die praktische Frage, ob es sich wirklich lohnt, kostenintensiv Entwickler Wochen oder gar Monate mit der Entwicklung eines leistungsfähigen LSTMs zu beschäftigen. Die Entwicklungszeit ist dabei bedeutend länger als bei einem traditionellen GARCH-Modell. Eine Kosten-Nutzen-Analyse sollte durchgeführt werden, um sicherzustellen, dass sich die Vorteile des LSTMs materialisieren lassen.[113]

---

[110] Vgl. Géron (2023), S. 266.
[111] Vgl. MaRisk vom 29.06.2023, AT 4.3.5.
[112] Keras ist eine Open-Source-Bibliothek, die eine benutzerfreundliche und hochproduktive Schnittstelle für die Entwicklung von KNN durch andere Bibliotheken (u.a. Tensorflow) für Deep-Learning in Python bietet.
[113] Vgl. insb. die Ergebnisse von LSTM und EGARCH(1,1) mit t-Verteilung.

Ein weiteres Problem bei der Entwicklung eines leistungsfähigen LSTMs ist das Fehlen von Standards. Derzeit gibt es keine standardisierten Datensätze und Benchmarks, an denen sich die Ergebnisse der Netze messen lassen. Zwar gibt es verschiedene wissenschaftliche Arbeiten, die die Prognosegüte von LSTMs untersucht haben, jedoch verwenden alle unterschiedliche Datensätze, Märkte und Zeiträume, was den Vergleich der Ergebnisse erschwert. Ein Modell, dass in einem Szenario überlegen ist, kann in einem anderen völlig versagen. Wenn ein Kreditinstitut eigene Benchmarks entwickeln würde, um die Transparenz und Nachvollziehbarkeit zu erhöhen, so steigen die Entwicklungszeit und die damit verbundenen Kosten der Implementierung.[114]

Selbst wenn es einer Bank gelingt, all die zuvor genannten Herausforderungen zu meistern, so ist die Nutzung der KNN für Mitarbeiter ohne spezielle IT-Expertise wahrscheinlich zu komplex. Dieses Problem ließe sich auf drei Wegen lösen: Erstens könnten die Finanzmitarbeiter ihre Anforderungen an die Bedienung des Netzes an die Entwickler weitergeben, um die Benutzerfreundlichkeit zu erhöhen. Zweitens könnten die Finanzmitarbeiter selbst eine Programmiersprache lernen, um das Netz ohne Frontend zu bedienen. Beide Ansätze sind entweder sehr zeitaufwendig oder herausfordernd. Drittens könnten neue Mitarbeiter mit vorhandener Expertise eingestellt werden. Es wird deutlich, dass das beste Modell keine Wertschöpfung generieren wird, wenn seine Nutzer es nicht adäquat verwenden können.[115]

Die Stabilität und Robustheit von Modellen ist ein weiterer kritischer Faktor. Schon geringfügige Änderungen an den Hyperparametern können erhebliche Auswirkungen auf die Leistung des Modells haben. Das Vertrauen auf Standardparameter und die Vernachlässigung einer gründlichen Hyperparameter-Optimierung kann die Modellleistung signifikant beeinträchtigen. Darüber hinaus ist es wichtig, genügend Hyperparameter zu berücksichtigen und diese durch bewährte Methoden zu optimieren − anstatt sie nur manuell zu justieren.

Die Optimierung der Hyperparameter ist jedoch ein zeitaufwendiger Prozess, der je nach Komplexität des Netzwerks viel Zeit in Anspruch nehmen kann. Beispielsweise hat die Berechnung der verschiedenen Kombinationen von versteckten Neuronen und Lags pro Parameter in dem verhältnismäßig einfach aufgebauten LSTM dieses Beitrags allein drei Stunden in Anspruch genommen. Diese zeitlichen Anforderungen steigen mit zunehmender Komplexität des Modells und der Anzahl der zu optimierenden Hyperparameter weiter an.[116]

---

[114] Vgl. Ahmed / Bahador (2018), S. 20f.; Vitali (2019), S. 71f.
[115] Vgl. Chen et al. (2023), S. 215.
[116] Die Zeitangabe basiert auf der Berechnungsdauer mit einer NVIDIA L4 Tensor Core GPU.

Das Training Künstlicher Neuronaler Netze ist wesentlich rechenintensiver als die Schätzung ökonometrischer Modelle. Moderne Deep-Learning-Algorithmen, die erfolgreich sehr tiefe KNN trainieren, können mehrere Wochen benötigen, um vollständig von Grund auf trainiert zu werden. Im Vergleich dazu benötigen traditionelle Modelle, wie einfache KNN, deutlich weniger Zeit, oft nur wenige Minuten bis Stunden, um trainiert zu werden. Der benötigte Rechenaufwand für ein KNN hängt dabei einerseits stark vom Umfang der Daten und andererseits von der Tiefe und Komplexität des Netzwerks ab.[117]

Um eine bessere Effizienz zu gewährleisten und den Zeitaufwand zu minimieren, wechseln viele Entwickler zu leistungsstarken Multi-Core-GPUs und ähnlichen Verarbeitungseinheiten. Diese verbesserten Prozessoren sind jedoch kosten- und stromverbrauchsintensiver. Rechenleistung ist aktuell ein knappes Gut und damit teures Gut. Kosten und ein erheblicher Energieverbrauch stellen somit eine zusätzliche Hürde für die Nutzung von komplexen KNN dar.[118]

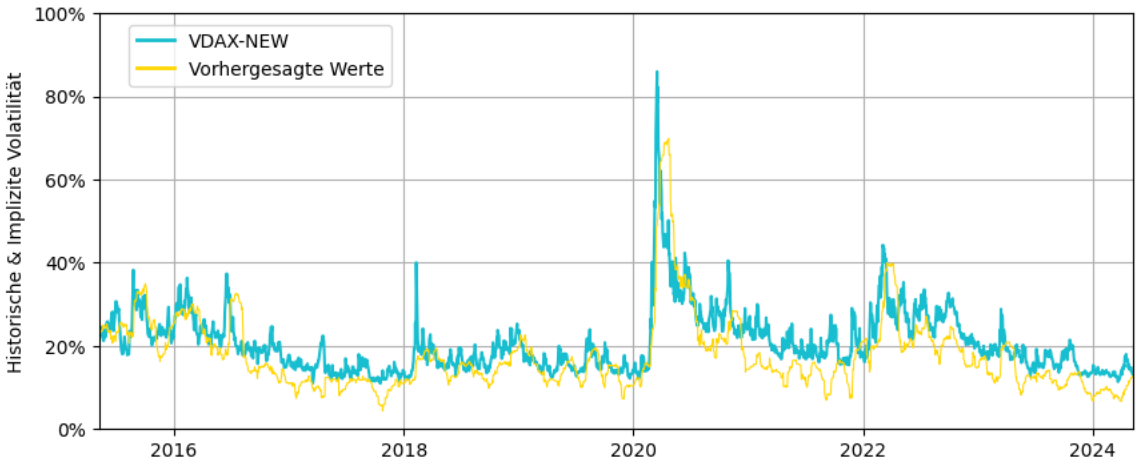

Abb. 23: Vorhergesagte LSTM-Werte und VDAX-NEW
Quelle: Eigene Darstellung mit Daten der Onvista Media

Im Weiteren leidet das LSTM in der Konfiguration dieses Beitrags unter dem Problem verzögerter Prognosen. Wenn lediglich die logarithmische Rendite und die historische 30-Tage Volatilitäten als Eingaben verwendet werden, ist die vom LSTM vorhergesagte Volatilität im Vergleich zur impliziten Volatilität verzögert. Dieser in Abb. 23 dargestellte Umstand führt dazu, dass die Prognose im Wesentlichen rückblickend ist und lediglich dem aktuellen Trend folgt. Einer der Gründe findet sich in der Effizienzmarkthypothese, die besagt, dass der Kapitalmarkt schnell auf alle verfügbaren Informationen reagiert, die effizient in die Preisbildung

[117] Vgl. Thompson et al. (2020), S. 1.
[118] Vgl. Cottier (2023), Absatz 1.

integriert werden können. Dementsprechend kann das trainierte LSTM mit nur 2 Merkmalen nicht alle, am Tag t verfügbaren Marktinformationen abdecken. Die Einführung zusätzlicher tagesaktueller Informationsquellen wie bspw. Stimmungsdaten und Nachrichten könnte dieses Problem potenziell verringern. Aktuell erfordert die Integration solcher zusätzlichen Daten in das Training neuronaler Netze komplexe Methoden, für die es noch keine standardisierte Vorgehensweise gibt.[119]

Eine grundsätzliche Schwäche von Künstlichen Neuronalen Netzen liegt im Datenhunger. Während zeitreihenökonometrische Verfahren wie GARCH auch mit relativ wenigen Daten (z. B. bei kurzen Zeitreihen) noch in sparsamer Parametrisierung angewandt werden können, setzt das Training eines KNN hinreichend viele Beobachtungswerte voraus.[120] Damit ein allgemeingültiges akzeptables Ergebnis berechnet werden kann, bedarf es umfangreicher Trainingsdaten.

Zudem erlaubt der parametrische Ansatz ökonometrischer Modelle das Prüfen von Kapitalmarkthypothesen wie dem diskutierten Leverage-Effekt, wohingegen neuronale Netze einen Black-Box-Ansatz darstellen und in der empirischen Finanzmarktanalyse im Wesentlichen auf ihren Einsatz als Prognoseinstrumente beschränkt sind. Ein trainiertes neuronales Netz stellt eine nichtlineare Funktion der Ausgabewerte in Abhängigkeit der Eingabewerte dar – ohne das zwingende Vorhandensein eines tiefergehenden Verständnisses von Ursache und Wirkungsbeziehungen.

[119] Vgl. Audrino et al. (2020), S. 334f.
[120] Vgl. Heaton et al. (2016), S. 18f.

7 Zusammenfassung und Ausblick

In diesem Beitrag wurden traditionelle ökonometrische Modelle zur Volatilitätsprognose mit modernen Deep-Learning-Ansätzen verglichen. Im Fokus standen dabei die symmetrischen und asymmetrischen GARCH-Modelle und das LSTM-Netzwerk. Diese wurden methodisch skizziert und im empirischen Teil hinsichtlich ihrer Prognosegenauigkeit für die zukünftige 30-Tage historische Volatilität verglichen.

Die empirischen Ergebnisse zeigen, dass das LSTM-Netzwerk sowohl In-sample als auch Out-of-sample genauere Ein-Schritt-Prognosen für den deutschen Aktienmarkt liefert als ein Durchschnitt aus GARCH-Modellen. Zudem konnte das LSTM besser auf neue Daten verallgemeinern, was es für Prognosen besonders geeignet macht. Zusätzlich wurden die Hyperparameter auf weitere Märkte außerhalb von Deutschland angewendet und es konnten ähnlich gute Ergebnisse festgestellt werden. Die Überlegenheit der LSTM-Netze lässt sich durch ihre Fähigkeit erklären, nicht-lineare Beziehungen zwischen Eingangs- und Ausgangsvariablen zu erkennen. Trotz dieser Vorteile bleibt die Intransparenz der LSTM-Netze ein Nachteil, der insbesondere im regulierten Bankensektor eine große Herausforderung darstellt.

Die empirische Überlegenheit des LSTM für die Volatilitätsprognose bietet einen erheblichen strategischen Vorteil im Bereich des Optionshandels und der dynamischen Absicherung von Exposures. Durch eine akkuratere Volatilitätsprognose, wie sie Künstliche Neuronale Netze wie das LSTM ermöglichen, könnte das Volatility-Risk-Premium reduziert und Optionen marktgerechter bepreist werden. Als Konsequenz würden auch die Kosten für den Delta-Hedge fallen.

Künftige methodische Weiterentwicklungen sollten sich ebenso darauf konzentrieren, die Transparenz und Erklärbarkeit von Deep-Learning-Verfahren zu verbessern, um deren Einsatz in der Praxis zu erleichtern. Die Entwicklung von Standards und Benchmarks der Prognosegüte könnte ebenfalls zur Akzeptanz und Implementierung im Finanzsektor beitragen. Insgesamt zeigen die Ergebnisse dieser empirischen Untersuchung das Potenzial von Künstlichen Neuronalen Netzen wie das LSTM als vielversprechende Alternative zu traditionellen zeitreihenökonometrischen Modellen in der Volatilitätsprognose.

Literaturverzeichnis

Ahmed, War / Bahador, Mehrdad (2018): The accuracy of the LSTM model for predicting the S&P 500 index and the difference between prediction and backtesting, https://kth.diva-portal.org/smash/get/diva2:1213449/FULLTEXT01.pdf.

Ang, Ing-Chea / Israelov, Roni / Sullivdan, Rodney / Tummala, Harsha (2018): Understanding Volatility Risk Premium, https://www.aqr.com/Insights/Research/White-Papers/Understanding-the-Volatility-Risk-Premium.

Audrino, Francesco / Sigrist, Fabio / Ballinari, Daniele (2020): The impact of sentiment and attention measures on stock market volatility, in: International Journal of Forecasting, Vol. 36, S. 334 - 357.

Auer, Ludwig von (2023): Ökonometrie - Eine Einführung, 8. Auflage, Springer Gabler.

Backhaus, Klaus / Erichson, Bernd / Gensler, Sonja / Weiber, Rolf / Weiber, Thomas (2021): Multivariate Analysemethoden - Eine anwendungsorientierte Einführung, 16. Auflage, Springer Gabler.

Bärlocher von Küsnacht, Jürg (1992): GARCH-Prozesse als Modelle für Devisenkurse, Winthertur.

Barz, Till / Nastansky, Andreas (2024): Herausforderungen des finanziellen Risikomanagements: Eine empirische Untersuchung des Value at Risk-Ansatzes in Stresssituationen, in: Statistische Diskussionsbeiträge, Nr. 57.

Becker-Carus, Christian / Wendt, Mike (2017): Allgemeine Psychologie - Eine Einführung, 2. Auflage, Springer.

Bera, Anil K / Higgins, Mattew L. (1993): ARCH models: properties, estimation and testing, in: Journal of Economic Surveys, Vol. 7(4), S. 305-366.

Bloss, Michael (2020): Financial Engineering - Strategien, Bewertungen und Risikomanagement, 4. Auflage, De Gruyter Oldenbourg.

Bollerslev, Tim (1986): Generalized Autoregressive Conditional Heteroscedasticity, in: Journal of Econometrics, Vol. 31, S. 307-327.

Box, George E. P. et al. (2016): Time Series Analysis - Forecasting and Control, 5. Auflage, Wiley.

Boyle, Patrick / McDougall, Jesse (2019): Trading and Pricing Financial Derivatives - A Guide to Futures, Options, and Swaps, De Gruyter.

Brooks, Chris (2008): Introductory Econometrics for Finance, 2. Auflage, Cambridge University Press.

Brown, Monika (2024): The Evolution of AI in Finance, https://www.chicagobooth.edu/review/evolution-ai-finance.

Bruns, Christoph / Meyer-Bullerdiek, Frieder (2020): Professionelles Portfoliomanagement: Aufbau, Umsetzung und Erfolgskontrolle strukturierter Anlagestrategien, 6. Auflage, Schäffer-Poeschel.

Bundesanstalt für Finanzdienstleistungsaufsicht (2023): MaRisk - Mindestanforderungen an das Risikomanagement, i.d.F. v. 29.06.2023, Rundschreiben 05/2023 (BA).

Cai, Weiting / Liu, Kaiyang / Lu, Keming (2023): Forecasting Daily Stock Volatility: A Comparison between GARCH and Recurrent Neuro-networks, in: BCP Business & Management, Vol. 38, S. 427-436.

Chen, Weisi / Hussain, Walayat / Cauteruccio, Francesco / Zhang, Xu (2023): Deep Learning for Financial Time Series Prediction: A State-of-the-Art Review of Standalone and Hybrid Models, in: CMES-Computer Modeling in Engineering & Sciences, Vol. 139(1), S. 187-224.

Cottier, Ben (2023): Trends in the Dollar Training Cost of Machine Learning Systems, https://epochai.org/blog/trends-in-the-dollar-training-cost-of-machine-learning-systems.

Degiannakis, Stavros / Xekalaki, Evdokia (2004): Autoregressive Conditional Heteroskedasticity (ARCH) Models: A Review, in: Quality Technology and Quantitative Management, Vol. 1(2), S. 271-324.

Demeterfi, Kresimir / Derman, Emanuel / Kamal, Michael / Zou, Joseph (1999): More Than You Ever Wanted To Know About Volatility Swaps - But Less Than Can Be Said, in: Goldman Sachs Quantitative Strategies Research Notes.

Deutsche Börse (2007): Leitfaden zu den Volatilitätsindizes der Deutschen Börse, 4. Auflage, Frankfurt am Main.

Deutsche Börse (2018): Leitfaden zu den Volatilitätsindizes der Deutschen Börse, 7. Auflage Frankfurt am Main.

Deutsche Börse (2015): DAX-Index – Benchmark und Barometer für die deutsche Wirtschaft.

DWS (2024): Börsencrashs im Rückblick: Vorsicht vor dem Fluchtreflex, https://www.dws.de/informieren/anlagethemen/aktien/historische-boersencrashs/.

Elman, Jeffrey L. (1990): Finding structure in time, in: Cognitive Science, Vol. 14(2), S. 179-211.

Engle, Robert F. (1982): Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation, in: Econometrica, Vol. 50, S. 987-1007.

Franke, Jürgen / Härdle, Wolfgang / Hafner, Christian (2003): Einführung in die Statistik der Finanzmärkte, 2. Auflage, Springer.

Ganguly, Santanu (2021): Quantum Machine Learning: An Applied Approach - The Theory and Application of Quantum Machine Learning in Science and Industry, Apress Berkeley.

German, Mark B. / Klass, Michael J. (1980): On the Estimation of Security Price Volatility from Historical Data, in: Journal of Business, Vol. 53(1), S. 67-78.

Géron, Aurélien (2023): Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow - Concepts, Tools, and Techniques to Build Intelligent Systems, 3. Auflage, O'Reilly.

Goodfellow, Ian / Bengio, Yoshua / Courville, Aaron (2016): Deep Learning, Cambridge University Press.

Greene, William H. (2019): Econometric Analysis, 8. Auflage, Pearson.

Guo, Qinghua / Jin, Shichao / Li, Min / Yang, Qiuli / Xu, Kexin / Ju, Yuanzhen / Zhang, Jing / Xuan, Jing / Liu, Jin / Su, Yanjun / Xu, Qiang / Liu, Yu (2020): Application of deep learning in ecological resource research: Theories, methods, and challenges, in: Science China Earth Sciences, Vol. 63(10), S. 1457-1474.

Hartung, Joachim (2009): Statistik: Lehr- und Handbuch der angewandten Statistik, 15. Auflage, Oldenbourg.

Haykin, Simon (2009): Neural Networks and Learning Machines, 3. Auflage, Pearson.

Heaton J.B. / Polson, Nicholas / Witte, Jan Hendrik (2016): Deep Learning in Finance, https://arxiv.org/abs/1602.06561

Hochreiter, Sepp / Schmidhuber, Jürgen (1997): Long Short-Term Memory, in: Neural Computation, Vol. 9(8), S. 1735-1780.

Hodjat, Babak (2015): The AI Resurgence: Why Now?, https://www.wired.com/insights/2015/03/ai-resurgence-now/#:~:text=For%20starters%2C%20the%20infrastructure%20speed,the%20shape%20of%20cloud%20services.

Hull, John C. (2015): Options, Futures, and Other Derivatives, 9. Auflage, Pearson.

Jiang, George J. / Tian Yisong S. (2007): Extracting Model-Free Volatility from Option Prices: An Examination of the VIX Index, in: Journal of Derivates, Vol. 14(3), S. 1-26.

Kraft, Dieter (1988): A software package for sequential quadratic programming, Wiss. Berichtswesen d. DFVLR.

Li, Kai / Weinbaum, David (2000): The Empirical Performance of Alternative Extreme Value Volatility Estimators, New York City University Press.

Markowitz, Harry M. (1952): Portfolio Selection, in: The Journal of Finance, Vol. 7, No. 1, 1952, S. 77-91.

Mauer, Annika / Nastansky, Andreas (2024): Empirische Analyse des Zusammenhangs zwischen Rendite und impliziter Volatilität am deutschen Aktienmarkt, in: Statistische Diskussionsbeiträge, Nr. 58.

McCulloch, Warren S. / Pitts, Walter (1943): A Logical Calculus Of The Ideas Immanent In Nervous Activity, in: Bulletin of Mathematical Biophysics, Vol. 5, 115-133.

Nastansky, Andreas (2012): Regression integrierter Zeitreihen am Beispiel einer kointegrierten Konsumfunktion, in: WiSt (Wirtschaftswissenschaftliches Studium), Heft 7, S. 367-373.

Nelson, Daniel B. (1991): Conditional Heteroskedasticity in Asset Returns: A New Approach, in: Econometrica, Vol. 59 (2), S. 347-370.

Neusser, Klaus / Wagner, Martin (2022): Zeitreihenanalyse in den Wirtschaftswissenschaften – Einführung und Grundlagen für den Einstieg in die aktuelle Forschung, 4. Auflage, Springer Gabler.

Nielsen, Michael (2018): Neural Networks and Deep Learning, Determination Press.

Parkinson, Michael (1980): The Extreme Value Method for Estimating the Variance of the Rate of Return, in: The Journal of Business, Vol. 53(1), S. 61-65.

Pesaran, Hashem M. (2015): Time Series and Panel Data Econometrics, Oxford University Press.

Poon, Ser-Huang / Granger, Clive W. J. (2003): Forecasting Volatility in Financial Markets: A Review, in: Journal of Economic Literature, Vol. 41(2), S. 478-539.

Schmidt, Friedrich / Trede, Mark (2006): Finanzmarktstatistik, Springer.

Schmitt, Christian (2012): Stochastische Volatilität, in: Schröder, Michael (Hrsg.): Finanzmarktökonometrie – Basistechniken, Fortgeschrittene Verfahren, Prognosemodelle, 2. Auflage, Schäffer-Poeschel.

Sheppard, Kevin (2021): Univariate volatility modeling, bootstrapping, multiple comparison procedures and unit root tests, https://arch.readthedocs.io/en/latest/index.html.

Shumway, Robert H. / Stoffer, David S. (2017): Time Series Analysis and Its Applications - With R Examples, 4. Auflage, Springer.

STOXX (2023): DAX Equity Index Methodology Guide, https://www.stoxx.com/document/News/2023/October/DAX%20Equity%20Index%20Methodology%20Guide_20231002.pdf.

Sydsaeter, Knut / Hammond, Peter / Strom, Arne / Carvajal, Andrés (2018): Mathematik für Wirtschaftswissenschaftler Basiswissen mit Praxisbezug, 5. Auflage, Pearson.

Tallau, Christian (2012): Zum Zusammenhang von DAX-Rendite und dem Volatilitätsindex VDAX-NEW, http://www.quantil-consulting.de/downloads/Zum-Zusammenhang-von-DAX-Renditen-und-dem-Volatilitaetsindex-VDAX-NEW.pdf.

Thompson, Neil C. / Greenewald, Kristjan / Lee, Keeheon / Manso, Gabriel F. (2020): The Computational Limits Of Deep Learning, in: MIT Initiative On The Digital Economy Research Brief, Vol. 4, S. 1-4.

Vitali, Greta (2019): Forecasting Stock Index Volatility: a comparison between GARCH and LSTM models, http://dspace.unive.it/bitstream/handle/10579/15933/868008-1231506.pdf?sequence=2.

Vollmer, Karl-Heinz / Bol, Georg / Nakhaeizadeh, Golamreza (1994): Finanzmarktanwendungen neuronaler Netze und ökonometrischer Verfahren, Physica Verlag.

Wagh, Akash (2022): Gradient Descent and its Types, https://www.analyticsvidhya.com/blog/2022/07/gradient-descent-and-its-types/.

Zakoian, Jean-Michel (1994): Threshold Heteroscedastic Models, in: Journal of Economic Dynamics and Control, Vol. 18, S. 931-955.

Zhang, Aston / Lipton, Zachary C. / Li, Mu / Smola, Alexander J. (2023): Deep Dive into Deep Learning, Cambridge University Press.

UNIVERSITÄT POTSDAM

STATISTISCHE DISKUSSIONSBEITRÄGE

Herausgeber: Andreas Nastansky

Nr. 1	1995	Strohe, Hans Gerhard: Dynamic Latent Variables Path Models - An Alternative PLS Estimation -
Nr. 2	1996	Kempe, Wolfram. Das Arbeitsangebot verheirateter Frauen in den neuen und alten Bundesländern - Eine semiparametrische Regressionsanalyse
Nr. 3	1996	Strohe, Hans Gerhard: Statistik im DDR-Wirtschaftsstudium zwischen Ideologie und Wissenschaft
Nr. 4	1996	Berger, Ursula: Die Landwirtschaft in den drei neuen EU-Mitgliedsstaaten Finnland, Schweden und Österreich - Ein statistischer Überblick
Nr. 5	1996	Betzin, Jörg: Ein korrespondenzanalytischer Ansatz für Pfadmodelle mit kategorialen Daten
Nr. 6	1996	Berger, Ursula: Die Methoden der EU zur Messung der Einkommenssituation in der Landwirtschaft - Am Beispiel der Bundesrepublik Deutschland
Nr. 7	1997	Strohe, Hans Gerhard / Geppert, Frank: Algorithmus und Computerprogramm für dynamische Partial Least Squares Modelle
Nr. 8	1997	Rambert, Laurence / Strohe, Hans Gerhard: Statistische Darstellung transformationsbedingter Veränderungen der Wirtschafts- und Beschäftigungsstruktur in Ostdeutschland
Nr. 9	1997	Faber, Cathleen: Die Statistik der Verbraucherpreise in Rußland - Am Beispiel der Erhebung für die Stadt St. Petersburg
Nr. 10	1998	Nosova, Olga: The Atractiveness of Foreign Direct Investment in Russia and Ukraine - A Statistical Analysis
Nr. 11	1999	Gelaschwili, Simon: Anwendung der Spieltheorie bei der Prognose von Marktprozessen
Nr. 12	1999	Strohe, Hans Gerhard / Faber, Cathleen: Statistik der Transformation - Transformation der Statistik. Preisstatistik in Ostdeutschland und Rußland
Nr. 13	1999	Müller, Claus: Kleine und mittelgroße Unternehmen in einer hoch konzentrierten Branche am Beispiel der Elektrotechnik. Eine statistische Langzeitanalyse der Gewerbezählungen seit 1882
Nr. 14	1999	Faber, Cathleen: The Measurement and Development of Georgian Consumer Prices
Nr. 15	1999	Geppert, Frank / Hübner, Roland: Korrelation oder Kointegration – Eignung für Portfoliostrategien am Beispiel verbriefter Immobilienanlagen
Nr. 16	2000	Achsani, Noer Azam / Strohe, Hans Gerhard: Statistischer Überblick über die indonesische Wirtschaft
Nr. 17	2000	Bartels, Knut: Testen der Spezifikation von multinominalen Logit-Modellen
Nr. 18	2002	Achsani, Noer Azam / Strohe, Hans Gerhard: Dynamische Zusammenhänge zwischen den Kapitalmärkten der Region Pazifisches Becken vor und nach der Asiatischen Krise 1997
Nr. 19	2002	Nosova, Olga: Modellierung der ausländischen Investitionstätigkeit in der Ukraine
Nr. 20	2003	Gelaschwili, Simon / Kurtanidse, Zurab: Statistische Analyse des Handels zwischen Georgien und Deutschland
Nr. 21	2004	Nastansky, Andreas: Kurz- und langfristiger statistischer Zusammenhang zwischen Geldmengen- und Preisentwicklung: Analyse einer kointegrierenden Beziehung
Nr. 22	2006	Kauffmann, Albrecht / Nastansky, Andreas: Ein kubischer Spline zur temporalen Disaggregation von Stromgrößen und seine Anwendbarkeit auf Immobilienindizes
Nr. 23	2006	Mangelsdorf, Stefan: Empirische Analyse der Investitions- und Exportentwicklung des Verarbeitenden Gewerbes in Berlin und Brandenburg
Nr. 24	2006	Reilich, Julia: Return to Schooling in Germany
Nr. 25	2006	Nosova, Olga / Bartels, Knut: Statistical Analysis of the Corporate Governance System in the Ukraine: Problems and Development Perspectives
Nr. 26	2007	Gelaschwili, Simon: Einführung in die statistische Modellierung und Prognose
Nr. 27	2007	Nastansky, Andreas: Modellierung und Schätzung von Vermögenseffekten im Konsum

UNIVERSITÄT POTSDAM

STATISTISCHE DISKUSSIONSBEITRÄGE

Herausgeber: Andreas Nastansky

Nr. 28	2008	Nastansky, Andreas: Schätzung vermögenspreisinduzierter Investitionseffekte in Deutschland
Nr. 29	2008	Ruge, Marcus / Strohe, Hans Gerhard: Analyse von Erwartungen in der Volkswirtschaft mit Partial-Least-Squares-Modellen
Nr. 30	2009	Newiak, Monique: Prüfungsurteile mit Dollar Unit Sampling – Ein Vergleich von Fehlerschätzmethoden für Zwecke der Wirtschaftsprüfung: Praxis, Theorie, Simulation –
Nr. 31	2009	Ruge, Marcus: Modellierung von Stimmungen und Erwartungen in der deutschen Wirtschaft
Nr. 32	2009	Nosova, Olga: Statistical Analysis of Regional Integration Effects
Nr. 33	2009	Mangelsdorf, Stefan: Persistenz im Exportverhalten – Kann punktuelle Exportförderung langfristige Auswirkungen haben? -
Nr. 34	2009	Kbiladze, David: Einige historische und gesetzgeberische Faktoren der Reformierung der georgischen Statistik
Nr. 35	2009	Nastansky, Andreas / Strohe, Hans Gerhard: Die Ursachen der Finanz- und Bankenkrise im Lichte der Statistik
Nr. 36	2009	Gelaschwili, Simon / Nastansky, Andreas: Development of the Banking Sector in Georgia
Nr. 37	2010	Kunze, Karl-Kuno / Strohe, Hans Gerhard: Time Varying Persistence in the German Stock Market
Nr. 38	2010	Nastansky, Andreas / Strohe, Hans Gerhard: The Impact of Changes in Asset Prices on Real Economic Activity: A Cointegration Analysis for Germany
Nr. 39	2010	Kunze, Karl-Kuno / Strohe, Hans Gerhard: Antipersistence in German Stock Returns
Nr. 40	2010	Dietrich, Irina / Strohe, Hans Gerhard: Die Vielfalt öffentlicher Unternehmen aus der Sicht der Statistik - Ein Versuch, das Unstrukturierte zu strukturieren
Nr. 41	2010	Nastansky, Andreas / Lanz, Ramona: Bonuszahlungen in der Kreditwirtschaft: Analyse, Regulierung und Entwicklungstendenzen
Nr. 42	2010	Dietrich, Irina / Strohe, Hans Gerhard: Die Vermögenslage öffentlicher Unternehmen in Deutschland - Statistische Analyse anhand von amtlichen Mikrodaten der Jahresabschlüsse.
Nr. 43	2010	Ulbrich, Hannes-Friedrich: Höherdimensionale Kompositionsdaten – Gedanken zur grafischen Darstellung und Analyse -
Nr. 44	2011	Dietrich, Irina / Strohe, Hans Gerhard: Statistik der öffentlichen Unternehmen in Deutschland – Die Datenbasis
Nr. 45	2011	Nastansky, Andreas: Orthogonale und verallgemeinerte Impuls-Antwort-Funktionen in Vektor-Fehlerkorrekturmodellen
Nr. 46	2011	Dietrich, Irina / Strohe, Hans Gerhard: Die Finanzlage öffentlicher Unternehmen in Deutschland - Statistische Analyse amtlicher Mikrodaten der Jahresabschlüsse
Nr. 47	2011	Teitge, Jonas / Nastansky, Andreas: Interdependenzen in den Renditen DAX-notierter Unternehmen nach Branchen
Nr. 48	2011	Dietrich, Irina: Die Ertragslage öffentlicher Unternehmen in Deutschland - Statistische Analyse amtlicher Mikrodaten der Jahresabschlüsse -
Nr. 49	2011	Kauper, Benjamin / Kunze, Karl-Kuno: Modellierung von Aktienkursen im Lichte der Komplexitätsforschung
Nr. 50	2011	Nastansky, Andreas / Strohe, Hans Gerhard: Konsumausgaben und Aktienmarktentwicklung in Deutschland: Ein kointegriertes vektorautoregressives Modell
Nr. 51	2014	Nastansky, Andreas / Mehnert, Alexander / Strohe, Hans Gerhard: A Vector Error Correction Model for the Relationship between Public Debt and Inflation in Germany
Nr. 52	2019	Kauffmann, Albrecht / Nastansky, Andreas: Explorative Analyse der Preise von Einfamilienhäusern und Eigentumswohnungen in Deutschland
Nr. 53	2019	Nastansky, Andreas: Topologische Datenanalyse: Eine Einführung in die Persistente Homologie und Mapper

UNIVERSITÄT POTSDAM

STATISTISCHE DISKUSSIONSBEITRÄGE

Herausgeber: Andreas Nastansky

Nr. 54	2022	Kauffmann, Albrecht / Nastansky, Andreas: Regionale Mieten in Deutschland: Explorative Analyse der Mieten in der Wiedervermietung
Nr. 55	2022	Nastansky, Andreas: Gruppierung von Daten: Topologische Verfahren vs. Clusteranalyse
Nr. 56	2023	Nastansky, Andreas / Siris, Sarah: Risikoverbund zwischen Banken und Staaten: Eine empirische Analyse für den Euroraum
Nr. 57	2024	Barz, Till / Nastansky, Andreas: Herausforderungen des finanziellen Risikomanagements: Eine empirische Untersuchung des Value at Risk-Ansatzes in Stresssituationen
Nr. 58	2024	Mauer, Annika / Nastansky, Andreas: Empirische Analyse des Zusammenhangs zwischen Rendite und impliziter Volatilität am deutschen Aktienmarkt
Nr. 59	2025	Knuth, Nico / Nastansky, Andreas: Anwendung von Deep Learning in der Prognose der Volatilität des DAX: Ein Vergleich der Prognosegüte von GARCH und LSTM