Bei diesem Beitrag handelt es sich um einen wissenschaftlich begutachteten und freigegebenen Fachaufsatz („reviewed paper“).

doi.org/10.37544/1436-4980-2026-04-51
Datum der Einreichung: 12.01.2026
Datum der Annahme: 12.02.2026
Datum der Veröffentlichung: 26. Mai 2026

Ressourcenschonende Strategien für industrielle Produktionsumgebungen

Hyperparameteroptimierung für Lastprognose

A. Harman, A. Sauer

ZUSAMMENFASSUNG Die Studie untersucht, wie ressourcenschonende Strategien zur Hyperparameteroptimierung die Genauigkeit und die Laufzeit industrieller Lastprognosen beeinflussen. Mit einem Taguchi-Design wurden zwei Modelle des maschinellen Lernens mit verschiedenen vereinfachenden Verfahren, sogenannten Pruning- und Subsampling-Methoden, getestet. Zufälliges Subsampling auf 30 % der Daten und Hyperband-Pruning erzielten teils bessere Prognosen bei deutlich geringerem Rechenaufwand.

STICHWÖRTER
Energieeffizienz, Künstliche Intelligenz (KI)

Efficient hyperparameter optimization for forecasts

ABSTRACT The study investigates the impact of low-fidelity strategies for hyperparameter optimization on the accuracy and computational costs of industrial load forecasting. Using a Taguchi design, two machine learning models were evaluated under different pruning and subsampling methods. Random subsampling of 30 % of the data combined with Hyperband pruning yielded equal or better predictive performance while significantly reducing computational resources.

1 Einleitung

Eine präzise Prognose des elektrischen Leistungsbedarfs in industriellen Produktionsumgebungen ist Voraussetzung für zahlreiche Energiemanagementanwendungen, darunter Demand-Side-Management, Lastspitzenreduktion und die zeitliche Steuerung flexibler Verbraucher zur Maximierung des Eigenverbrauchs. Die zunehmende Integration erneuerbarer Energiequellen wie Wind und Sonne erhöht die Komplexität der Prognosen zusätzlich, da ihre volatile Einspeisung im Vergleich zu fossilen Erzeugern mehr Unsicherheit in die Netzlast einbringt [1, 2].

Um diese steigende Komplexität zu bewältigen, werden Lastprognosen zunehmend mit Methoden des maschinellen Lernens (ML) erstellt. Jedes ML-System erfordert die Festlegung von Hyperparametern [3], beispielsweise der Anzahl der Bäume oder des Split-Kriteriums in Random Forests [4] sowie der Lernrate und der Anzahl versteckter Schichten in neuronalen Netzen [5]. Im Gegensatz zu Modellparametern wie Regressionsgewichten, die während des Trainings aus den Daten gelernt werden, müssen Hyperparameter vor dem Training festgelegt werden, da sie die Modellarchitektur bestimmen [6]. Die Wahl geeigneter Hyperparameter ist herausfordernd, da die optimale Konfiguration datensatzabhängig ist und daher für jede Anwendung neu angepasst werden muss. Es existiert keine allgemeingültige mathematische Regel, stattdessen basiert die Hyperparameteroptimierung (HPO) überwiegend auf Versuch und Irrtum [7].

Die Durchführung der HPO ist gerade bei fein zeitaufgelösten industriellen Lastdaten äußerst rechenintensiv. Modelle werden wiederholt trainiert, häufig über Kreuzvalidierung (cross-validation, CV) evaluiert und auf großen Datensätzen optimiert. Hier setzen ressourcenschonende Strategien an, die darauf abzielen, den Aufwand der Modellbewertung durch Vereinfachungen zu reduzieren. Durch diese Vereinfachungen lässt sich ein Trade-Off zwischen Rechenaufwand und Prognosegüte herstellen. Vereinfachte Modellbewertung kann beispielsweise bedeuten, dass ein Modell nur auf einem reduzierten Datenabschnitt trainiert wird, statt auf dem gesamten Datensatz [8].

In diesem Beitrag wird der Einfluss verschiedener ressourcenschonender Strategien zur Beschleunigung der HPO auf Rechenzeit und Modellleistung bei kurzfristigen Lastprognosen mit einem Prognosehorizont von vier Stunden systematisch analysiert. Ziel ist es zu untersuchen, inwieweit ressourcenschonende Strategien eine Beschleunigung der HPO ermöglichen, ohne signifikante Beeinträchtigung der Prognosegenauigkeit.

Die untersuchten Strategien umfassen eine unterschiedliche Anzahl an Teilmengen (Folds) von CV und verschiedene Pruning- sowie Subsampling-Methoden mit jeweils fest definierten, unterschiedlichen Teilmengengrößen. Die Verwendung fester Teilmengengrößen ist darin begründet, dass viele bestehende Verfahren dynamisch variierende Teilmengen verwenden, während feste Teilmengen eine bessere Kontrolle über die Rechenzeit erlauben. Der Einsatz fester Teilmengen in der HPO ist vor allem in Szenarien vorteilhaft, in denen:

- Rechenressourcen geteilt, budgetiert oder generell begrenzt sind (etwa in akademischen oder industriellen Umgebungen),
- umfangreiche Benchmarking-Experimente mit vielen Modellen durchgeführt werden müssen,
- der Fokus eher auf dem Leistungsvergleich von Modellen als auf der Identifikation eines einzelnen Modells liegt.

1.1 Verwandte Arbeiten

Der grundlegendste Ansatz zur HPO ist das manuelle Tuning. Nach der Erstellung eines ML-Modells werden Hyperparameterwerte iterativ basierend auf Erfahrung, Heuristiken oder der Auswertung vorheriger Ergebnisse ausgewählt. Dieser Prozess wird so lange wiederholt, bis die angestrebte Prognosegüte erreicht ist oder das zur Verfügung stehende Zeitbudget aufgebraucht ist. Trotz seiner Einfachheit ist manuelles Tuning für viele Anwendungen ungeeignet, da viele Hyperparameter berücksichtigt werden müssen, Modelle komplex aufgebaut sind, Evaluierungen zeitaufwendig erfolgen, nichtlineare Interaktionen zwischen Hyperparametern auftreten und die Reproduzierbarkeit begrenzt ist [3].

Eine systematische Alternative ist das vollfaktorielle Design, bei dem sämtliche Kombinationen von Hyperparametern evaluiert werden. Obwohl dieser Ansatz alle geplanten Kombinationen abdeckt, wächst die Anzahl der erforderlichen Evaluierungen exponentiell mit der Anzahl der zu untersuchenden Hyperparameter, sodass er bei mehr als wenigen Hyperparametern nicht mehr sinnvoll anwendbar ist [3].

Um die Skalierbarkeit zu erhöhen, hat sich die Zufallssuche als weit verbreitete Alternative etabliert. Hierbei werden Konfigurationen zufällig ausgewählt, bis ein vorgegebenes Budget in Form einer maximalen Anzahl von Modellevaluierungen ausgeschöpft ist. Trotz seiner Einfachheit kann die Zufallssuche mit deutlich weniger Evaluierungen eine vergleichbare Modellleistung erreichen und schneidet damit erheblich besser ab als vollfaktorielle Designs [3].

Bayes'sche Optimierung bietet einen effizienteren Ansatz, indem die Beziehungen zwischen Hyperparametern und Modellleistung mittels probabilistischer Surrogat-Modelle, wie etwa Gaußschen Prozessen, modelliert werden. Dadurch wird das Hyperparameterproblem als Optimierungsaufgabe formuliert. Die Bayes'sche Optimierung definiert eine Prior-Verteilung über die Zielfunktion, aktualisiert diese sequenziell zu einer Posteriori-Verteilung, sobald neue Evaluierungen vorliegen, und wählt mit einet Akquisitionsfunktion die jeweils vielversprechendste nächste Konfiguration aus [9].

Weitere Ansätze umfassen evolutionäre Algorithmen wie genetische Algorithmen. Diese erfordern jedoch eine zusätzliche Eigenparametrisierung und weisen aufgrund der größeren Anzahl notwendiger Auswertungen höheren Rechenaufwand auf als Bayes'sche Optimierung [10].

Zur Reduktion der Rechenkosten der HPO wurden verschiedene Algorithmen vorgeschlagen. Pruning-Methoden brechen früh ab, um Evaluierungen zu beenden, die voraussichtlich keine konkurrenzfähige Leistung erzielen. Der Asynchronous Successive Halving Algorithm (ASHA) weist beispielsweise einer großen Menge an Konfigurationen ein kleines Budget zu, evaluiert diese, behält nur den leistungsstärksten Anteil bei und erhöht anschließend das Budget für die verbliebenen Konfigurationen. Dieser Prozess wird wiederholt bis das maximale Budget pro Konfiguration erreicht ist [11]. Der Hyperband-Algorithmus formuliert HPO als ein rein exploratives, nichtstochastisches Infinite-Armed-Bandit-Problem. Im Gegensatz zu ASHA, das eine feste Anzahl von Konfigurationen voraussetzt, berücksichtigt Hyperband adaptiv mehrere mögliche Konfigurationsanzahlen bei konstantem Gesamtbudget [12].

Weitere Methoden nutzen Datenpartitionierung zur Beschleunigung der HPO. „TrimTuner" [13] evaluiert Kandidatenkonfigurationen auf Teilsegmenten der Datensätze, um den Trainingsaufwand zu reduzieren. Diese Auswertungen werden mit Bayes'scher Optimierung und Transfer-Learning-Techniken genutzt, um die Modellleistung auf dem vollständigen Datensatz zu prognostizieren und Konfigurationen auszuwählen, die eine hohe Modellleistung erwarten lassen und vorgegebene Ressourcenrestriktionen einhalten. Der „Fabolas"-Algorithmus [14] behandelt die Datensatzgröße als zusätzliche kontinuierliche Eingabe und modelliert Verlustfunktion und Evaluierungskosten gemeinsam als Funktionen der Hyperparameter und der Teilmengengröße. Typischerweise beginnt der Algorithmus mit sehr kleinen Teilmengen und erweitert diese nur, wenn ein Informationsgewinn erwartet wird.

Koskela et al. [15] schlagen ein Poisson-Subsampling-Schema vor, bei dem Hyperparameter auf zufälligen Teilmengen abgestimmt und die resultierenden Modelle anschließend auf größeren Datensätzen trainiert werden. Gleichzeitig wird die Lernrate gemäß der Datengröße skaliert, während andere Hyperparameter konstant bleiben. *Jin* [16] stellt ein subsampling-basiertes Verfahren zur Schätzung der Hyperparameterrelevanz vor, bei dem diese mittels Subsampling-Prozeduren approximiert werden, um effizientes Tuning zu ermöglichen, indem die wichtigsten Hyperparameter auf dem vollständigen Datensatz priorisiert werden.

1.2 Beitrag der Arbeit

Obwohl verschiedene Algorithmen zur Beschleunigung der HPO existieren, basieren die meisten auf einer dynamischen Anpassung der Datensatzgröße während der Optimierung. Es fehlt an einer systematischen Untersuchung, wie sich Subsampling-Methoden mit fester Teilmengengröße, vor allem mit für Zeitreihen geeigneten Ansätzen, in Kombination mit Pruning-Methoden auf Modellleistung und Rechenaufwand innerhalb eines Bayes'schen Optimierungsrahmens auswirken.

Die Beiträge dieser Arbeit sind:

- eine systematische Analyse verschiedener HPO-Designentscheidungen, einschließlich der Anzahl an CV-Folds, der eingesetzten Pruning-Methoden (Hyperband, ASHA, kein Pruning) sowie fester Teilmengengrößen mit unterschiedlichen Anteilen und für Zeitreihen geeigneten Subsampling-Methoden, eingebettet in einen Bayes'schen Optimierungsansatz und angewendet auf zwei verschiedene ML-Modelle zur elektrischen Lastprognose;
- eine empirische Untersuchung des Einflusses dieser Methoden auf Rechenzeit und Prognosegenauigkeit;
- eine praxisorientierte Empfehlung für ein HPO-Setup, das die Rechenzeit deutlich reduziert, ohne die Prognosegenauigkeit wesentlich zu beeinträchtigen, geeignet für Benchmarking-Szenarien sowie industrielle Lastprognoseanwendungen.

2 Methodik

Ziel der Methodik ist es, die in Kapitel 1 beschriebenen HPO-Strategien (Anzahl der CV-Folds, Pruning- und Subsampling-Methoden, Teilmengengrößen) systematisch zu variieren und deren Einfluss auf Rechenzeit und Prognosegüte zu quantifizieren. Die folgenden Abschnitte beschreiben die zur Umsetzung dieser Analyse verwendeten Modelle, Daten, Suchräume und Optimierungsverfahren.

2.1 Datenvorverarbeitung

Um die Prognosegüte zu verbessern, wurde der Datensatz mit Zeitreihendaten um exogene Variablen erweitert, darunter meteorologische Merkmale (wie Außentemperatur, Bewölkungsgrad) sowie kalenderbasierte Variablen wie Feiertage und Wochenenden, da diese Faktoren das Lastverhalten beeinflussen. Zusätzlich wurden verzögerte Werte (Lags) der Prädiktoren integriert, um zeitliche Abhängigkeiten explizit abzubilden und die Autokorrelation in den Modellen auszunutzen.

Vor dem Modelltraining wurden übliche Vorverarbeitungsschritte durchgeführt. Fehlende Werte wurden anhand der Werte derselben Stunde der Vorwoche aufgrund der starken wöchentlichen Autokorrelation imputiert. Kontinuierliche Merkmale wurden anschließend standardisiert.

Obwohl die Experimente auf einem einzigen Datensatz durchgeführt wurden, wurden zwei zeitliche Auflösungen (5 min und 15 min) verwendet, um die Robustheit und Generalisierbarkeit der HPO-Strategien zu bewerten. Daten mit höherer Auflösung (5 min) erfassen kurzfristige Schwankungen und erhöhen das Datenvolumen, während niedrigere Auflösungen (15 min) zu geglätteten Zeitreihen mit geringerer Rechenlast führen. Dieses Design ermöglicht die Bewertung, ob die Ergebnisse hinsichtlich der Effizienz und Leistungsfähigkeit von ressourcenschonenden HPO-Strategien über unterschiedliche in der Lastprognose gebräuchliche zeitliche Auflösungen hinweg konsistent bleiben.

Abschließend wurde der Datensatz zeitlich im Verhältnis 80:20 in separate Trainings- und Testdatensätze unterteilt, um zu verhindern, dass Informationen aus dem Testdatensatz während der HPO genutzt werden und dadurch eine verzerrte Leistungsbewertung entsteht.

2.2 Prognosemodelle

Für die Lastprognose wurden zwei ML-Modelle mit dem Python Framework Darts [17] implementiert, die der Analyse von HPO-Strategien dienen: ein baumbasiertes und ein neuronales Netzwerk. Beide Modelle werden häufig in der Zeitreihenprognose eingesetzt und verfügen über eine Vielzahl von Hyperparametern, deren Abstimmung die Modellleistung maßgeblich beeinflusst.

Extreme Gradient Boosting (XGB) ist eine recheneffiziente, regularisierte Gradient-Boosting-Methode, die einen effektiven Ausgleich zwischen Modellkomplexität und Generalisierung bietet und eine effiziente Optimierung der Modellanpassung erlaubt [18].

Das Temporal Convolutional Neural Network (TCN) ist ein neuronales Modell, welches für Zeitreihenprognosen entwickelt wurde. Durch den Einsatz dilatierter Faltungen kann das Modell lange historische Abhängigkeiten erfassen und Sequenzen variabler Länge modellieren, ohne dabei Informationsleckage von der Zukunft in die Vergangenheit zuzulassen [19].

2.3 Hyperparameteroptimierung: Verfahren und Strategien

Der folgende Abschnitt fasst die wesentlichen Komponenten der HPO zusammen. In dieser Studie wurden vier zentrale HPO-Faktoren untersucht: (1) die Anzahl der CV-Folds, (2) die eingesetzten Pruning-Methoden (Hyperband, ASHA, kein Pruning), (3) die Subsampling-Methoden (deterministisch, zufällig, kein Subsampling) und (4) die verwendeten Teilmengengrößen (10 %, 30 %, 50 %). Diese Faktoren wurden systematisch variiert, um ihren Einfluss auf Rechenzeit und Prognosegüte zu isolieren.

2.3.1 Optimierungsverfahren

Für die HPO wurde eine Bayes'sche Optimierung unter Verwendung des Optuna-Frameworks [20] eingesetzt. Die Bayes'sche Optimierung dient hierbei als übergeordneter Suchrahmen, innerhalb dessen die ressourcenschonenden Strategien systematisch variiert wurden. Die Anzahl der Optimierungsläufe und Trainingsepochen wurde bewusst niedrig gehalten, um die Rechenlast zu begrenzen. Die Anzahl der Optimierungsläufe wurde auf 20 begrenzt, was mit den Iterationszahlen vergleichbarer Studien übereinstimmt [21, 22]. Obwohl eine höhere Anzahl von Optimierungsläufen potenziell zu besseren Ergebnissen führen kann, wurde die gewählte Obergrenze für die hier durchgeführte vergleichende Analyse als ausreichend erachtet, um Unterschiede zwischen den untersuchten HPO-Strategien konsistent zu erfassen.

Ebenso wurde die Anzahl der Trainingsepochen auf 10 festgelegt, was geringer ist als bei einem Training bis zur vollständigen Konvergenz. Frühere Arbeiten zeigen jedoch, dass bereits wenige Epochen ausreichen können, um Modelle mit hohem Potenzial von solchen mit geringem Potenzial zu unterscheiden [8].

2.3.2 Hyperparameter-Suchräume

Die Hyperparameter-Suchräume wurden so definiert, dass ein Gleichgewicht zwischen Recheneffizienz und Modellkapazität gewährleistet ist. Beispielsweise wurden die Anzahl der Bäume in XGB sowie die Dilation-Base und die Batchgröße im TCN auf begrenzte Werte festgelegt, um den Rechenaufwand zu reduzieren, während gleichzeitig genügend Flexibilität erhalten bleibt, um den Vier-Stunden-Vorhersagehorizont adäquat abzubilden.

Tabelle 1 fasst die untersuchten Suchräume für beide Modelle zusammen.

2.3.3 Ressourcenschonende Strategien

Im Rahmen der ressourcenschonenden HPO analysiert diese Studie den Einfluss verschiedener Subsampling-Anteile und Subsampling-Methoden, unterschiedlicher Pruning-Methoden sowie der Anzahl an CV-Folds auf Prognosegüte und Rechenaufwand.

Die Anzahl der CV-Folds wurde zwischen 5 und 10 variiert. Die Subsampling-Anteile wurden auf 10 %, 30 % und 50 % festgelegt. Die Teilmengen wurden entweder mittels deterministischer oder zufälliger Subsampling-Methoden ausgewählt, oder es wurde vollständig auf Subsampling verzichtet. Beim deterministischen Ansatz wurde das Teilsegment aus einem vordefinierten mittleren Abschnitt des Datensatzes entnommen, da dieser Abschnitt typische tägliche und wöchentliche Lastmuster aufweist und damit die Gesamtstruktur gut abbildet. Beim zufälligen Ansatz wurde ein aufeinanderfolgendes Datenfenster der jeweiligen Länge zufällig ausgewählt. Beide Methoden gewährleisten eine kontinuierliche Datenfolge, was für Zeitreihenmodelle essenziell ist.

Als Pruning-Methoden kamen der ASHA-Algorithmus, Hyperband sowie ein Baseline-Ansatz ohne Pruning zum Einsatz.

Tabelle 1 Untersuchte Hyperparameter-Suchräume für XGB und TCN.

XGB			TCN		
Hyperparameter	min	max	Hyperparameter	min	max
Maximale Baumtiefe	3	8	Kernelgröße	2	4
Lernrate	0,005	0,3	Dilationsbasis	2	3
Anzahl der Bäume	50	400	Anzahl Filter	8	32
Mindestgewicht eines Blatts	1	5	Anzahl Ebenen	1	3
Subsampling-Anteil (Zeilen)	0,6	1	Dropout-Anteil	0	0,4
Subsampling-Anteil (Features)	0,6	1	Batchgröße	32	48
Gamma	0	5	Lernrate	5,00E-04	5,00E-02

2.4 Experimentaldesign

Die große Anzahl möglicher Faktorkombinationen erzeugt einen großen Suchraum, dessen vollständige Untersuchung zu aufwändig wäre. Daher wurde als Versuchsdesign das Taguchi-L24-Orthogonalarray eingesetzt, um die systematische Analyse aller untersuchten HPO-Strategien mit vertretbarem Rechenaufwand zu ermöglichen. Dieses Design erlaubt die Untersuchung einer Vielzahl von Parametern bei deutlich reduzierter Anzahl erforderlicher Experimente. Taguchi-Designs können Faktoren mit unterschiedlichen Stufenzahlen abbilden und werden in den Ingenieurwissenschaften in experimentellen Studien breit eingesetzt. Obwohl Interaktionseffekte nicht vollständig geschätzt werden können, wird diese Einschränkung in vielen Anwendungsfeldern, vor allem bei der Parameteroptimierung, als unkritisch angesehen [23–25]. Das Design wurde mit dem R-Paket „DOE.base“ [26] generiert. Die Zuordnung der Faktorstufen im Taguchi-L24-Orthogonalarray ist in **Tabelle 2** dargestellt.

2.5 Evaluationssetup

Die HPO wurde ausschließlich auf dem Trainingsdatensatz unter Verwendung von CV durchgeführt, während ein unabhängiger Testdatensatz über den gesamten Optimierungsprozess hinweg vollständig unberührt blieb. Nachdem für jede experimentelle Konfiguration die besten Hyperparameter identifiziert wurden, wurden die Modelle erneut auf dem vollständigen Trainingsdatensatz trainiert und anschließend einmal auf dem Testdatensatz evaluiert, um die finale Leistungsbewertung zu erhalten.

Für jede Experimentkonfiguration wurde die Optimierungsdauer aufgezeichnet. Die Modellleistung wurde anhand des Mean Absolute Error (MAE) bewertet, definiert als

$$MAE = \frac{1}{n}\sum_{i=1}^{n} \left| \hat{Y}_i - Y_i \right|$$

Die Verwendung einer einfachen und gut interpretierbaren Kennzahl erleichtert den Vergleich der HPO-Strategien.

MAE wurde als primäre Metrik gewählt, da alle Experimente auf einem einzigen Datensatz durchgeführt wurden und somit eine direkte Vergleichbarkeit der Einstellungen gegeben ist. Zudem liefert MAE eine intuitive und robuste Maßzahl für die durchschnittliche Abweichung zwischen vorhergesagten und tatsächlichen Werten in derselben physikalischen Einheit wie die Zielgröße. Im Gegensatz zum Mean Absolute Percentage Error (MAPE) bleibt MAE unempfindlich gegenüber Werten nahe Null, und im Gegensatz zum Mean Squared Error (MSE) werden große Fehler nicht überproportional stark gewichtet. Metriken wie R^2 oder erklärte Varianz wurden nicht berücksichtigt, da sie primär Korrelationen statt einer absoluten Prognosegenauigkeit quantifizieren.

Da in dieser Studie mehrere Zielgrößen gleichzeitig betrachtet werden, waren klassische Signal-to-Noise-Ratios (S/N), wie sie häufig in Taguchi-Experimenten zur Bestimmung optimaler Faktorstufen eingesetzt werden [27], ungeeignet. Stattdessen wurde ein zweistufiges modellbasiertes Entscheidungsverfahren angewendet.

Zunächst wurden zwei lineare gemischte Modelle (Linear Mixed-Effects Models, LMM) geschätzt, eines mit dem MAE und eines mit der Optimierungsdauer als Zielvariable. Die experimentellen Faktoren (Anzahl der CV-Folds, Pruning-Methode, Subsampling-Anteil und Subsampling-Methode) wurden als feste Effekte berücksichtigt. LMMs erlauben es, sowohl Korrelationen innerhalb als auch zwischen Gruppen abzubilden [28]. Die Kombination aus Prognosemodell (XGB oder TCN) und Datenfrequenz (5 min oder 15 min) wurde als Gruppierungsfaktor aufgenommen, um systematische Leistungsunterschiede zwischen den Modell-Frequenz-Kombinationen abzubilden.

Auf Basis der geschätzten LMMs wurden Vorhersagen für alle möglichen HPO-Konfigurationen (das heißt den vollständigen kombinatorischen Raum aller Faktorstufen) erzeugt. Für jede Konfiguration wurden danach Harrington-Wünschbarkeitswerte (Desirability-Werte) für MAE und für die Optimierungsdauer berechnet, wobei beide Zielgrößen als „je kleiner, desto besser“ behandelt wurden. Die Harrington-Desirability-Funktion ordnet jede Zielgröße einem Wertebereich von [0,1] zu, wobei höhere Werte eine höhere Wünschbarkeit repräsentieren [29]. Ein globaler Desirability-Index wurde danach als geometrisches Mittel der beiden Einzelwerte berechnet, wodurch Prognosegenauigkeit und Recheneffizienz gleichmäßig berücksichtigt werden.

Die HPO-Konfiguration mit dem höchsten globalen Desirability-Index wurde als bestes Setup identifiziert und anschließend wurde ein unabhängiger Bestätigungsdurchlauf durchgeführt, um diese Auswahl zu validieren. Dieses Vorgehen ist konzeptionell ähnlich dem Ansatz von *Abbas et al.* [23], bei dem ebenfalls Regressionsanalysen mit einem zusammengesetzten Desirability-Index verknüpft werden, um optimale Konfigurationen in multiobjektiven Optimierungsproblemen zu bestimmen.

Tabelle 2 Zuordnung der Faktorstufen im Taguchi-L24-Orthogonalarray.

Anzahl der CV-Folds	Pruning-Methode	Subsampling-Anteil	Subsampling-Methode
5	ASHA	10%	Zufällig
5	ASHA	30%	Deterministisch
5	ASHA	50%	Zufällig
5	ASHA	100%	Kein Subsampling
5	Hyperband	10%	Zufällig
5	Hyperband	30%	Deterministisch
5	Hyperband	50%	Zufällig
5	Hyperband	100%	Kein Subsampling
5	Kein Pruning	10%	Deterministisch
5	Kein Pruning	30%	Zufällig
5	Kein Pruning	50%	Deterministisch
5	Kein Pruning	100%	Kein Subsampling
10	ASHA	10%	Deterministisch
10	ASHA	30%	Zufällig
10	ASHA	50%	Deterministisch
10	ASHA	100%	Kein Subsampling
10	Hyperband	10%	Deterministisch
10	Hyperband	30%	Zufällig
10	Hyperband	50%	Deterministisch
10	Hyperband	100%	Kein Subsampling
10	Kein Pruning	10%	Zufällig
10	Kein Pruning	30%	Deterministisch
10	Kein Pruning	50%	Zufällig
10	Kein Pruning	100%	Kein Subsampling

2.6 Rechenumgebung

Die Experimente wurden auf einer Workstation mit einem „Intel Xeon Platinum 8268" CPU @ 2.90 GHz sowie zwei „Nvidia RTX A6000" GPUs mit jeweils 48 GB dediziertem Speicher durchgeführt. Um die Vergleichbarkeit der Rechenzeiten zwischen den verschiedenen HPO-Konfigurationen sicherzustellen, wurde die Parallelisierung während des Tuning-Prozesses deaktiviert. Dadurch wurde gewährleistet, dass Laufzeitunterschiede ausschließlich aus den getesteten HPO-Einstellungen resultieren und nicht durch unterschiedliche Ressourcenzuteilung beeinflusst werden.

3 Ergebnisse

Dieses Kapitel stellt die Ergebnisse der Studie vor. Dabei werden die Auswirkungen der getesteten HPO-Konfigurationen auf die Prognosegenauigkeit und die Optimierungsdauer sowie die kombinierte Bewertung mittels Harrington-Desirability-Werte dargestellt.

3.1 Datensatz

Der zugrunde liegende Datensatz stammt von einem deutschen Automobilzulieferer und umfasst einen Zeitraum von sechs Monaten in der zweiten Hälfte des Jahres 2023. Er enthält aggregierte Leistungsdaten eines Produktionsbereichs. Die Zeitreihe weist ausgeprägte zeitliche Muster auf, die tägliche und wöchentliche Produktionszyklen widerspiegeln.

Obwohl der Beobachtungszeitraum begrenzt ist, wurde er für die Zielsetzung dieser Studie als geeignet erachtet, da der Schwerpunkt auf dem Vergleich verschiedener HPO-Strategien lag und nicht auf dem Erzielen langfristiger Prognosegenauigkeit.

3.2 Einfluss der HPO-Einstellungen auf die Optimierungsdauer

Bild 1 zeigt die Verteilung der Optimierungszeiten für die einzelnen HPO-Konfigurationen. Beim XGB-Modell lag die maximale Optimierungsdauer unter 6 Stunden, während sie beim TCN-Modell über 80 Stunden hinausging. Die höhere zeitliche

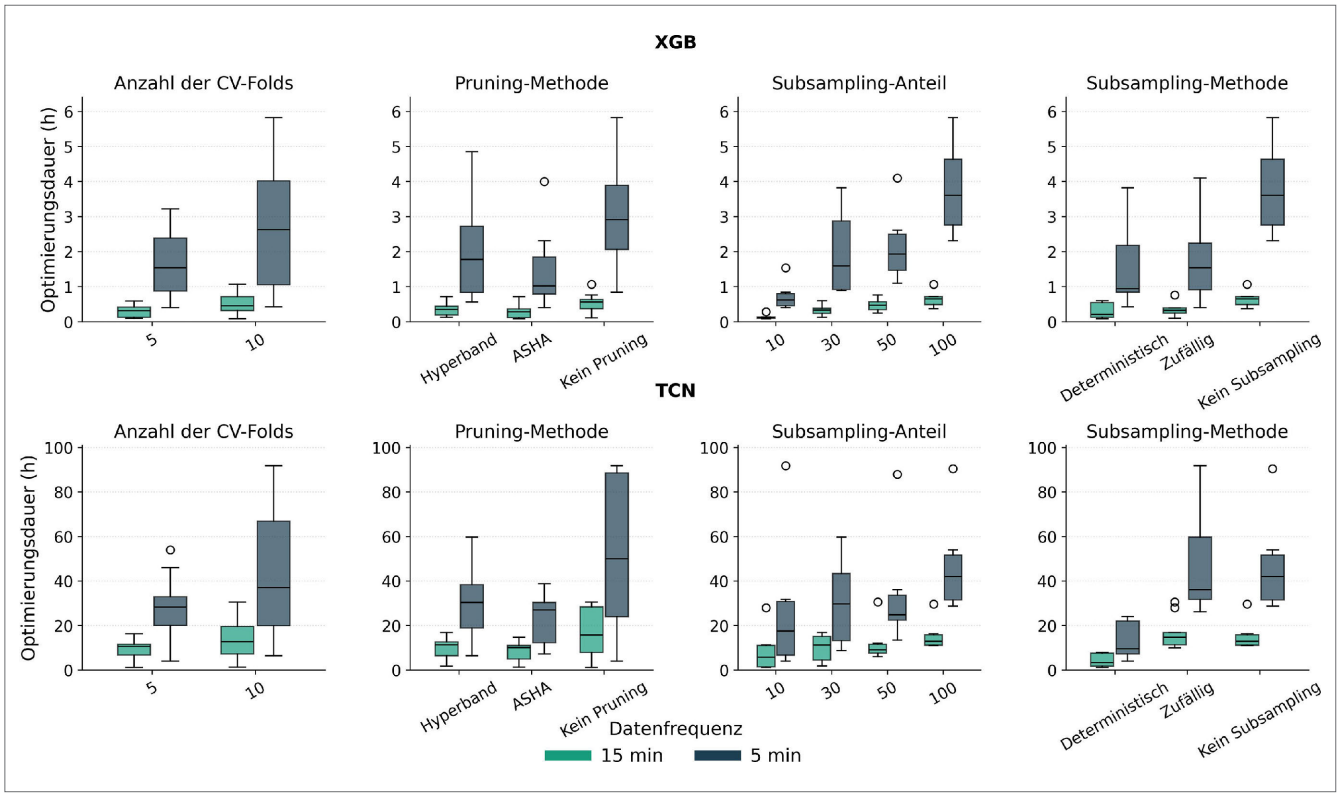

Bild 1 Optimierungsdauer für verschiedene Hyperparameteroptimierung (HPO)-Einstellungen. *Grafik: Fraunhofer IPA*

Auflösung (5 min) führte aufgrund des größeren Datenvolumens zu deutlich längeren Laufzeiten. Ebenso nahmen die Optimierungszeiten mit einer höheren Anzahl an CV-Folds zu.

Unter den zwei Pruning-Methoden erzielte ASHA die kürzesten Laufzeiten, gefolgt von Hyperband. Beide reduzierten die Optimierungsdauer deutlich im Vergleich zu Durchläufen ohne Pruning. Mit steigendem Subsampling-Anteil erhöhte sich die Optimierungsdauer erwartungsgemäß. Zwischen den beiden Subsampling-Methoden führte das deterministische Subsampling zu den kürzesten Optimierungszeiten.

3.3 Einfluss der HPO-Einstellungen auf die Prognosegenauigkeit

Bild 2 zeigt die normalisierten MAE-Werte für die einzelnen HPO-Designentscheidungen, skaliert auf den Bereich $[0,1]$, um Vergleichbarkeit und Vertraulichkeit sicherzustellen. Insgesamt fällt der Einfluss der HPO-Einstellungen auf die Prognosegenauigkeit geringer aus als auf die Optimierungsdauer.

Die Anzahl der CV-Folds zeigte nahezu keinen Effekt auf die Modellleistung. Obwohl Pruning- und Subsampling-Methoden die Optimierungszeit erheblich reduzierten, führten sie nicht zu einem konsistenten Verlust an Prognosegenauigkeit. Beide Pruning-Methoden erreichten eine Genauigkeit, die mit den Läufen ohne Pruning vergleichbar war, wobei letztere insbesondere beim XGB-Modell mit höherer zeitlicher Auflösung eine größere Interquartilspanne aufwiesen.

Die Verwendung der Subsampling-Methoden führte zu MAE-Werten, die denen des vollständigen Datensatzes entsprachen oder in einigen Konfigurationen sogar darunter lagen. Dies deutet darauf hin, dass reduzierte, aber gut gewählte Datenausschnitte vergleichbare Ergebnisse ermöglichen können.

3.4 Kombinierte Effekte der HPO-Einstellungen auf Optimierungsdauer und Prognosegenauigkeit

Die Harrington-Desirability-Werte für den vorhergesagten MAE sowie die Optimierungsdauer aller HPO-Konfigurationen sind in **Bild 3** dargestellt.

Insgesamt zeigte sich, dass HPO auf dem vollständigen Datensatz durchgehend zu niedrigeren Desirability-Werten für sowohl MAE als auch Optimierungsdauer führte. Dies weist darauf hin, dass die Nutzung des Gesamtdatensatzes keinen Genauigkeitsvorteil bietet, gleichzeitig aber die Laufzeit erheblich erhöht. Ebenso erzielten Konfigurationen ohne Pruning deutlich geringere Desirability-Werte hinsichtlich der Optimierungsdauer, da sie sehr rechenintensiv sind, ohne systematische Verbesserungen in der Prognosegüte zu liefern.

Die Wahl der Subsampling-Methode hatte einen starken Einfluss auf den Trade-Off zwischen Laufzeit und Prognosequalität. Deterministisches Subsampling reduzierte zwar die Optimierungsdauer deutlich, führte jedoch vor allem bei kleinen Subsampling-Anteilen oft zu niedrigen MAE-Desirability-Werten. Zufälliges Subsampling hingegen erzielte bereits bei 10–30 % der Daten sowohl die besten Prognoseergebnisse als auch die höchste Recheneffizienz und markierte damit den günstigsten Bereich hinsichtlich des globalen Desirability-Index.

Die Konfiguration mit dem höchsten globalen Desirability-Index kombinierte Hyperband-Pruning, zufälliges Subsampling mit 30 % der Daten sowie fünf CV-Folds. Diese Konfiguration wurde anschließend in einem unabhängigen Bestätigungsdurchlauf validiert. **Tabelle 3** fasst die relativen Verbesserungen von MAE und Optimierungsdauer im Vergleich zur Baseline, definiert als HPO auf dem vollständigen Datensatz ohne Pruning und mit fünf CV-Folds, zusammen.

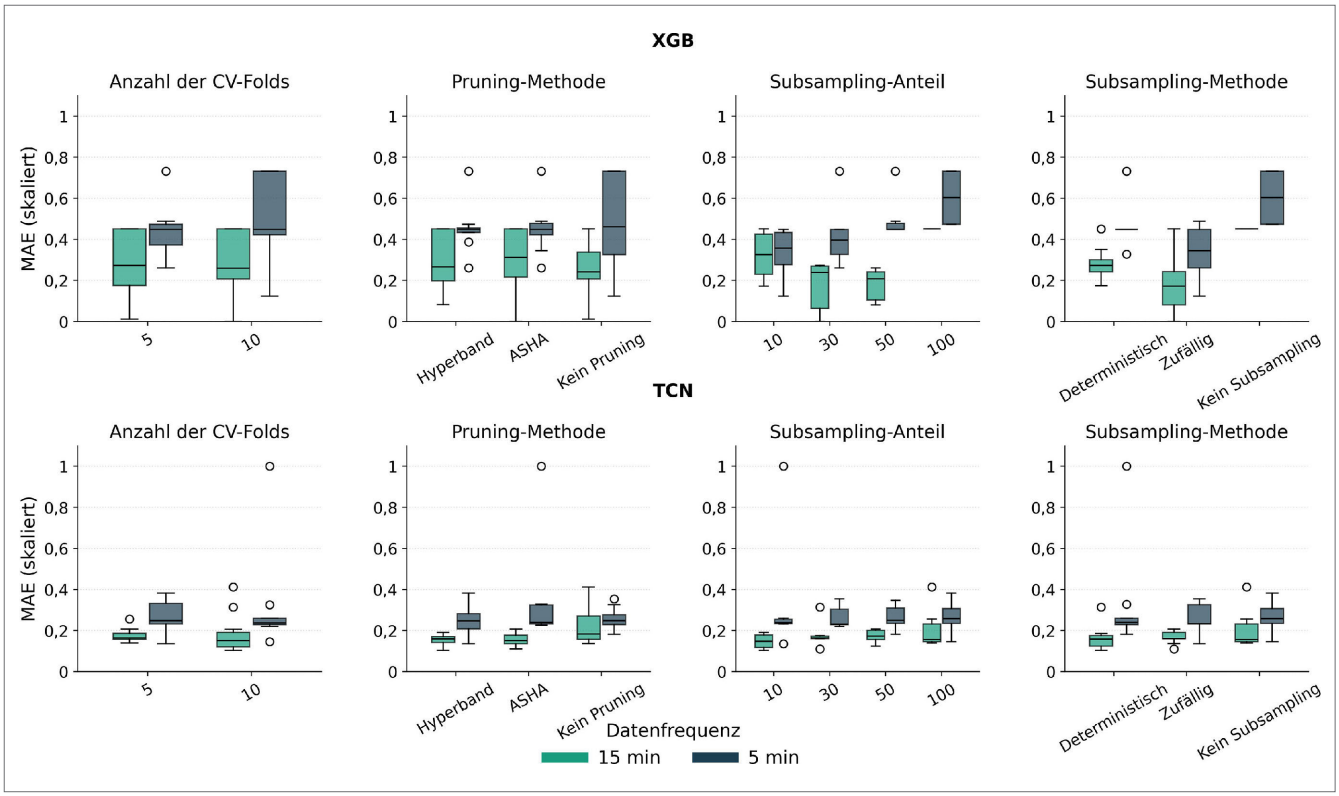

Bild 2 Mean Absolute Error (MAE) für verschiedene HPO-Einstellungen. *Grafik: Fraunhofer IPA*

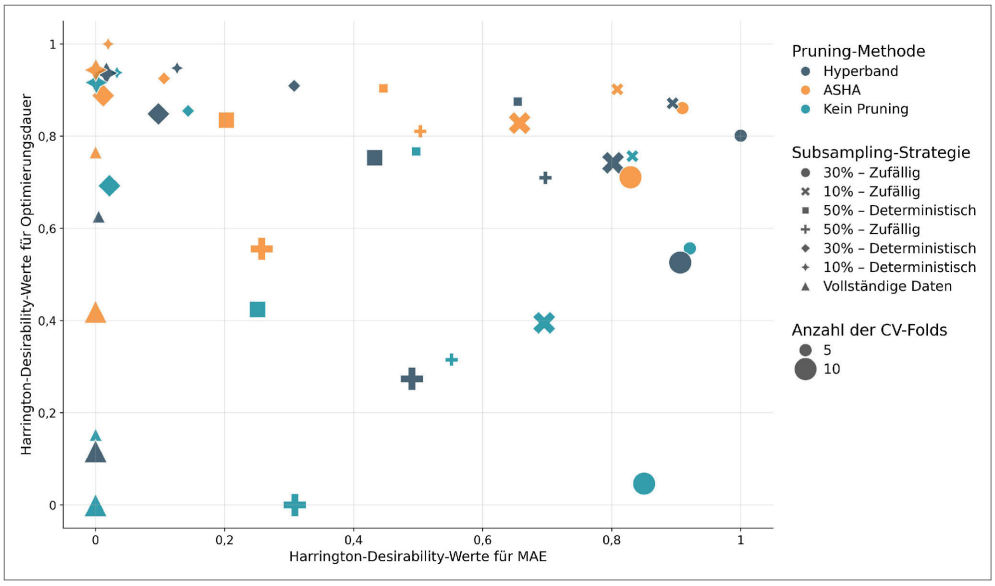

Bild 3 Harrington-Desirability-Werte für den vorhergesagten MAE und die Optimierungsdauer über alle HPO-Konfigurationen; höhere Werte entsprechen einer höheren Wunschbarkeit. *Grafik: Fraunhofer IPA*

Im Durchschnitt reduzierte die ausgewählte Konfiguration den MAE um 10,06 % für XGB und 2,06 % für TCN, während die Optimierungsdauer um 55,04 % (XGB) beziehungsweise 26,61 % (TCN) sank.

4 Diskussion

4.1 Einschränkungen

Obwohl die Studie systematische Erkenntnisse zu effizienten ressourcenschonenden Strategien für die Lastprognose liefert, sind mehrere Einschränkungen zu berücksichtigen.

Erstens basiert die Analyse auf einem einzigen Datensatz (wenn auch in zwei zeitlichen Auflösungen), was die Übertragbarkeit der Ergebnisse auf andere Anwendungsdomänen einschränkt. Auch wurden nur zwei Prognosemodelle untersucht. Für andere Modellfamilien, etwa transformerbasierte Architekturen oder statistische Baseline-Modelle, könnten die Resultate abweichen.

Zweitens wurden die Experimente unter begrenzten Rechenressourcen ausgeführt. Dies führte zu einer geringen Anzahl an Iterationen und Trainingsepochen. Obwohl dieser Ansatz durch frühere Arbeiten gestützt wird, schränkt er dennoch die Modellkomplexität und -ausprägung ein. Aus demselben Grund konnten

Tabelle 3 Verbesserung bei MAE und Optimierungsdauer durch die ausgewählte HPO-Konfiguration im Vergleich zur Baseline (vollständiger Datensatz, kein Pruning, 5-fach CV) für beide Modelle und Frequenzen.

Modell	Frequenz	MAE-Verbesserung	Reduktion der Optimierungsdauer
XGB	5 min	13,06%	58,55%
XGB	15 min	7,05%	51,53%
XGB-Durchschnitt		**10,06%**	**55,04%**
TCN	5 min	1,37%	30,47%
TCN	15 min	2,76%	22,75%
TCN-Durchschnitt		**2,06%**	**26,61%**
Durchschnitt		**6,06%**	**40,82%**

während der Optimierung nur begrenzte Hyperparameter-Suchräume untersucht werden.

Der Vorhersagehorizont war auf vier Stunden festgelegt, weshalb die Ergebnisse nicht unmittelbar auf sehr kurzfristige oder deutlich längerfristige Prognoseaufgaben übertragbar sind.

Methodisch bringt das verwendete Taguchi-Design weitere Einschränkungen mit sich. Im Gegensatz zum vollfaktoriellen Design können Interaktionseffekte zwischen Faktoren nicht vollständig erfasst werden, und es wird eine lineare Beziehung zwischen Faktoren und Zielgrößen angenommen, was in der Praxis nicht immer zutrifft [30]. Auch die im zweiten Schritt eingesetzten LMMs beruhen auf Linearitätsannahmen, sodass potenzielle nichtlineare Zusammenhänge möglicherweise nicht vollständig abgebildet werden [28].

Trotz dieser Einschränkungen liefert die Studie einen praktisch relevanten und recheneffizienten Rahmen zur Auswahl geeigneter HPO-Strategien und bildet eine solide Grundlage für zukünftige Forschungsarbeiten.

4.2 Ausblick

Die beschriebenen Einschränkungen weisen auf mehrere vielversprechenden Ansatzpunkte für zukünftige Forschungsarbeiten hin, welche die Aussagekraft und Generalisierbarkeit der Ergebnisse erweitern könnten.

Künftige Studien sollten mehrere Datensätze mit unterschiedlichen zeitlichen, saisonalen und betrieblichen Charakteristika einbeziehen und die Analyse auf ein breiteres Spektrum an Prognosemodellen ausdehnen.

Mit größeren Rechenressourcen könnten zudem umfangreichere Hyperparameter-Suchräume erkundet, mehr Iterationen und Trainingsepochen durchgeführt sowie weitere Vorhersagehorizonte untersucht werden, um besser zu verstehen, wie sich ressourcenschonende HPO-Konfigurationen in verschiedenen Prognoseaufgaben verhalten. Ebenso könnte der Einfluss alternativer Pruning- und Subsampling-Methoden detaillierter analysiert werden.

Aus methodischer Perspektive wäre es lohnend, das Taguchi-Design durch vollfaktorielle Designs oder adaptive Design-of-Experiments-Ansätze zu ersetzen, um Interaktionseffekte und nichtlineare Zusammenhänge besser erfassen zu können. Ebenso könnten modellbasierte Auswahlverfahren über LMMs hinaus erweitert werden, um komplexere Abhängigkeiten zu berücksichtigen.

Insgesamt würden diese Erweiterungen dazu beitragen, ein umfassenderes und generalisierbares Verständnis darüber zu entwickeln, wie ressourcenschonende HPO-Strategien sowohl die Prognosegenauigkeit als auch die Recheneffizienz in verschiedenen Lastprognoseszenarien beeinflussen.

5 Schlussfolgerung

Die Studie untersuchte den Einfluss von ressourcenschonenden Strategien zur HPO auf die Prognosegenauigkeit und den Rechenaufwand bei der Lastprognose. Mithilfe eines Taguchi-L24-Designs wurden zwei ML-Modelle (XGB und TCN) in verschiedenen Kombinationen von Pruning-Methoden, Subsampling-Anteilen und -Methoden sowie unterschiedlichen CV-Fold-Iterationsgrößen evaluiert. Anschließend wurden LMMs zur Vorhersage von MAE und Optimierungsdauer über alle möglichen HPO-Konfigurationen hinweg eingesetzt und die optimale Einstellung mithilfe von Harrington-Desirability-Werten bestimmt.

Die Konfiguration mit dem höchsten globalen Desirability-Index kombinierte Hyperband-Pruning, zufälliges Subsampling mit 30 % der Daten sowie fünf CV-Folds. Im Vergleich zur Baseline (vollständiger Datensatz, kein Pruning, fünf Folds) verbesserte diese Einstellung modellübergreifend im Mittel die MAE um 6,06 % und reduzierte gleichzeitig die Optimierungsdauer um 40,82 %.

Mehrere zentrale Erkenntnisse lassen sich daraus ableiten. Erstens können ressourcenschonende HPO-Strategien die Optimierungsdauer drastisch reduzieren, ohne die Prognosegenauigkeit zu verschlechtern. Die HPO mit vollständigem Datensatz zeigte durchgehend die geringere Desirability bei zugleich höchstem Rechenaufwand, was verdeutlicht, dass größere Datenmengen nicht zwangsläufig zu besseren Hyperparametern führen. Zweitens boten die Pruning-Methoden (Hyperband und ASHA) deutliche Beschleunigungen bei stabiler Genauigkeit. Drittens erwies sich zufälliges Subsampling als besonders effektiv: Schon 10–30 % der Daten reichten oft aus, um qualitativ hochwertige Hyperparameter zu identifizieren.

Im Vergleich zu konventionellen Vorgehensweisen wie manuellem HPO oder vollfaktorielle Design bieten die untersuchten ressourcenschonenden HPO-Strategien eine deutlich effizientere Alternative. Während traditionelle Ansätze sehr rechenintensiv sind, ermöglichen diese Verfahren eine schnellere Identifikation leistungsfähiger Hyperparameter mit deutlich reduziertem Aufwand. Dies zeigt das erhebliche praktische Potenzial von ressour-

censchonenden HPO-Strategien, vor allem in Umgebungen mit begrenzten Rechenressourcen oder in groß angelegten Benchmarking-Studien. Die Befunde belegen, dass sich erhebliche Rechenzeit einsparen lässt, ohne Einbußen in der Modellleistung. Damit sind effizientere und skalierbare Experimente in der Lastprognose und verwandten ML-Anwendungen möglich.

FÖRDERHINWEIS

Die Autoren danken dem Bundesministerium für Wirtschaft und Klimaschutz (BMWK) für die finanzielle Förderung und dem Projektträger Jülich (PtJ) für die Projektbetreuung im Rahmen des Projekts „FlexGUIde". Zudem danken wir dem Projektpartner für das Bereitstellen der Daten.

Literatur

[1] Paulus, M.; Borggrefe, F.: The potential of demand-side management in energy-intensive industries for electricity markets in Germany. Applied Energy 88 (2011) 2, pp.432–441, doi.org/10.1016/j.apenergy.2010.03.017

[2] Sharma, M.; Mittal, N.; Mishra, A. et al.: Survey of Electricity Demand Forecasting and Demand Side Management Techniques in Different Sectors to Identify Scope for Improvement. Smart Grids and Sustainable Energy 8 (2023) 2, #9, doi.org/10.1007/s40866–023–00168-z

[3] Hutter, F.; Kotthoff, L.; Vanschoren, J. (eds.): Automated Machine Learning: Methods, Systems, Challenges. Cham: Springer International Publishing 2019, doi.org/10.1007/978–3–030–05318–5

[4] Probst, P.; Wright, M. N.; Boulesteix, A.: Hyperparameters and tuning strategies for random forest. WIREs Data Mining and Knowledge Discovery 9 (2019) 3, pp.e1301, doi.org/10.1002/widm.1301

[5] Diaz, G. I.; Fokoue-Nkoutche, A.; Nannicini, G.; Samulowitz, H.: An effective algorithm for hyperparameter optimization of neural networks. IBM Journal of Research and Development 61 (2017) 4/5, pp. 9:1–9:11, doi.org/10.1147/JRD.2017.2709578

[6] Yang, L.; Shami, A.: On hyperparameter optimization of machine learning algorithms: Theory and practice. Neurocomputing 415 (2020), pp. 295–316, doi.org/10.1016/j.neucom.2020.07.061

[7] Andonie, R.: Hyperparameter optimization in learning systems. Journal of Membrane Computing 1 (2019) 4, pp. 279–291, https://doi.org/10.1007/s41965–019–00023–0

[8] Egele, R.; Guyon, I.; Sun, Y.; Balaprakash, P.: Is One Epoch All You Need For Multi-Fidelity Hyperparameter Optimization? ArXiv 2023, https://doi.org/10.48550/arXiv.2307.15422

[9] Wu, J.; Chen, X.-Y.; Zhang, H.; Xiong, L.-D.; Lei, H.; Deng, S.-H.: Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization. Journal of Electronic Science and Technology 17, (2019) 1, pp. 26–40, http://doi.org/10.11989/JEST.1674–862X.80904120

[10] Puentes G., D. E.; Barrios H., C. J.; Navaux, P. O. A.: Hyperparameter Optimization for Convolutional Neural Networks with Genetic Algorithms and Bayesian Optimization. 2022 IEEE Latin American Conference on Computational Intelligence (LA-CCI). Montevideo, Uruguay: IEEE 2022, pp. 1–5, https://doi.org/10.1109/LA-CCI54402.2022.9981104

[11] Li, L.; Jamieson, K.; Rostamizadeh, A. et al.: A System for Massively Parallel Hyperparameter Tuning. ArXiv 20218, https://doi.org/10.48550/arXiv.1810.05934

[12] Li, L.; Jamieson, K.; DeSalvo, G. et al.: Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. ArXiv 2018, https://doi.org/10.48550/arXiv.1603.06560

[13] Mendes, P.; Casimiro, M.; Romano, P. et al.: TrimTuner: Efficient Optimization of Machine Learning Jobs in the Cloud via Sub-Sampling. 28th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS). Nice, France, IEEE 2020, pp. 1–8, https://doi.org/10.1109/MASCOTS50786.2020.9285971

[14] Klein, A.; Falkner, S.; Bartels, S. et al.: Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets. ArXiv 2016, https://doi.org/10.48550/arXiv.1605.07079

[15] Koskela, A.; Kulkarni, T.: Practical Differentially Private Hyperparameter Tuning with Subsampling. ArXiv 2023, https://doi.org/10.48550/arXiv.2301.11989

[16] Jin, H.: Hyperparameter Importance for Machine Learning Algorithms. ArXiv 2022, https://doi.org/10.48550/arXiv.2201.05132

[17] Herzen, J.; Lässig, F.; Piazzetta, S. G. et al.: Darts: User-friendly forecasting for Time Series. Journal of Machine Learning Research 23 (2022) 124, pp. 1–6, jmlr.org/papers/v23/21–1177.html

[18] Chen, T.; Guestrin, C.: XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016, pp. 785–794, https://doi.org/10.1145/2939672.2939785

[19] Bai, S.; Kolter, J. Z.; Koltun, V.: An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. ArXiv 2018, https://doi.org/10.48550/arXiv.1803.01271

[20] Akiba, T.; Sano, S.; Yanase, T. et al.: Optuna: A Next-generation Hyperparameter Optimization Framework. ArXiv 2019, https://doi.org/10.48550/arXiv.1907.10902

[21] Victoria, A. H.; Maragatham, G.: Automatic tuning of hyperparameters using Bayesian optimization. Evolving Systems 12 (2021) 1, pp. 217–223, https://doi.org/10.1007/s12530–020–09345–2

[22] Vincent, A. M.; Jidesh, P.: An improved hyperparameter optimization framework for AutoML systems using evolutionary algorithms. Scientific Reports 13 (2023) 1, #4737, https://doi.org/10.1038/s41598-023-32027-3

[23] Abbas, A. T.; Ragab, A. E.; Benyahia, F. et al.: Taguchi Robust Design for Optimizing Surface Roughness of Turned AISI 1045 Steel Considering the Tool Nose Radius and Coolant as Noise Factors. Advances in Materials Science and Engineering 2018 (2018) 1, #2560253, https://doi.org/10.1155/2018/2560253

[24] Amir, B.; Gale, Y.; Sadot, A. et al.: Study on the effects of manufacturing parameters on the dynamic properties of AlSi10Mg under dynamic loads using Taguchi procedure. Materials & Design 223 (2022), #111125, https://doi.org/10.1016/j.matdes.2022.111125

[25] Khaw, J. F. C.; Lim, B. S.; Lim, L. E. N.: Optimal design of neural networks using the Taguchi method. Neurocomputing 7 (1995) 3, pp. 225–245, https://doi.org/10.1016/0925–2312(94)00013-I

[26] Groemping, U.: DoE.base: Full Factorials, Orthogonal Arrays and Base Utilities for DoE Packages. 2009, pp. 1.2–5, https://doi.org/10.32614/CRAN.package.DoE.base

[27] Mitra, A.: The Taguchi method. WIREs Computational Statistics 3 (2011) 5, pp. 472–480, https://doi.org/10.1002/wics.169

[28] Wu, L.: Mixed Effects Models for Complex Data. University of British Columbia 2009, www.stat.ubc.ca/~lang/Book/mybook.pdf

[29] Quirante, T.; Sebastian, P.; Ledoux, Y.: A trade-off function to tackle robust design problems in engineering. Journal of Engineering Design 24 (2013) 1, pp. 64–81, https://doi.org/10.1080/09544828.2012.691160

[30] Hisam, M. W.; Dar, A. A.; Elrasheed, M. O.; Khan, M. S.; Gera, R.; Azad, I.: The Versatility of the Taguchi Method: Optimizing Experiments Across Diverse Disciplines. Journal of Statistical Theory and Applications 23 (2024) 4, pp. 365–389, https://doi.org/10.1007/s44199-024-00093-9

Anna Harman, M.Sc.
anna.harman@ipa.fraunhofer.de

Prof. Dr.-Ing. Dipl.-Kfm. Alexander Sauer

Fraunhofer-Institut für Produktionstechnik
und Automatisierung IPA
Nobelstr. 12, 70569 Stuttgart
www.ipa.fraunhofer.de

Prof. Dr.-Ing. Dipl.-Kfm. Alexander Sauer

Institut für Energieeffizienz in der Produktion (EEP)
der Universität Stuttgart
Nobelstr. 12, 70569 Stuttgart
www.eep.uni-stuttgart.de