ChatGPT

Überblick: ML-Modelle und hybride Ansätze für 3D-Druck-Resin-Formulierungsoptimierung

Hintergrund und Problemstellung

Die Formulierung von photopolymerisierbaren 3D-Druck-Harzen, die keramische Partikel als Füllstoffe enthalten, ist ein komplexes Optimierungsproblem in der Materialwissenschaft. Eine typische Resin-Formulierung umfasst mehrere Komponentenklassen – z.B. Keramikpartikel, Monomere, Oligomere, Photoinitiatoren, Dispergiermittel, Thixotropiemittel (Verdicker), UV-Absorber und nicht-reaktive Additive – deren genaue Zusammensetzung die **Druckbarkeit** (etwa Viskosität, Sedimentationsstabilität) und die **Materialeigenschaften** des gedruckten Bauteils (wie Aushärtetiefe, Oberflächenglätte, Härte, Festigkeit usw.) bestimmt. Die Suche nach optimalen Kombinationen dieser Komponenten ist schwierig, da die Anzahl möglicher Rezepturen sehr groß ist und die Beziehungen zwischen Zusammensetzung und Eigenschaften hochgradig nichtlinear und oft nicht **a priori** bekannt sind. Traditionell erfolgt die Rezepturentwicklung durch empirisches Trial-and-Error; maschinelles Lernen (ML) bietet jedoch einen datengestützten Ansatz, um aus vorhandenen (wenn auch wenigen) historischen Daten und eingebrachtem Expertenwissen Muster abzuleiten und bessere Rezepturvorschläge zu generieren [1] [2] .

In unserem Anwendungsfall gibt es zwei zentrale Zielstellungen: **(1) Vorhersage**: Gegeben eine bestimmte Kombination von Materialkomponenten und deren Anteilen, sollen die resultierenden Druckergebnisse bzw. Materialeigenschaften (etwa Aushärtetiefe, Härte, Oberflächenqualität etc.) vorhergesagt werden. **(2) Optimierung**: Basierend auf gewünschten Zielwerten für solche Eigenschaften soll umgekehrt eine oder mehrere optimale Materialkombinationen vorgeschlagen werden, die diese Ziele erfüllen. Dabei ist herausfordernd, dass nur wenige historische Formulierungs-Datensätze verfügbar sind und diese zudem heterogen und teils verrauscht oder unvollständig vorliegen. Der Einbezug von **synthetischen Daten** (z.B. durch Simulationen oder generative Modelle) und **Expertenwissen** (z.B. chemische Regeln, Materialontologien) kann hier entscheidend zur Verbesserung beitragen. Prozessparameter des Druckvorgangs (Schichthöhe, Belichtungszeit etc.) bleiben zunächst konstant oder außerhalb des Modells, um die Komplexität zu reduzieren – sie könnten jedoch in Zukunft als zusätzliche Eingaben ergänzt werden.

Kategorien von ML-Ansätzen für Materialvorhersage und -optimierung

Für das genannte Problem kommen verschiedenste Typen von ML-Algorithmen in Frage. Im Folgenden werden **klassische**, **probabilistische**, **tiefe neuronale** und **symbolisch/wissensbasierte** Verfahren hinsichtlich ihrer Eignung diskutiert, insbesondere mit Blick auf kleine vs. große Datenmengen und auf Vorhersage- vs. Optimierungsaufgaben.

Klassische ML-Verfahren: Unter klassischen Ansätzen versteht man hier etablierte Algorithmen des überwachten Lernens wie lineare/nichtlineare Regressionsmodelle, Entscheidungsbäume und Ensemble-Methoden (Random Forest, Gradient Boosting) sowie Kernel-Methoden (Support Vector Machines). Solche Algorithmen sind in der Materialinformatik weit verbreitet, da sie oft schon mit relativ wenigen Daten robust funktionieren und weniger aufwändige Hyperparameteranpassungen erfordern

[3] [4]. Beispielsweise wurden Entscheidungsbaum-Ensembles (Random Forest, XGBoost etc.) erfolgreich eingesetzt, um mechanische Eigenschaften von 3D-Druck-Polymerkompositen vorherzusagen [4] [5]. Diese Modelle können nichtlineare Wechselwirkungen zwischen Rezepturkomponenten abbilden und bleiben auch bei moderaten Datenmengen stabil, da Techniken wie **Bagging** und **Boosting** Overfitting entgegenwirken. SVMs mit geeigneten Kernel-Funktionen sind ebenso für begrenzte Datensätze geeignet, da der Kernel-Trick komplexe nichtlineare Zusammenhänge in höherdimensionale Räume abbilden kann, während das Margin-Konzept der SVM für gute **Generalisation** auch bei kleinem Datensatz sorgt [6] [7]. Ein Beispiel aus der Literatur ist die Arbeit von Nasrin et al., die mit einem kleinen Datensatz hochgefüllter Harzformulierungen (unterschiedliche Partikelgrößenverteilungen) ein SVM-ähnliches künstliches Neuronales Netz (ANN) trainierten, um die *Druckbarkeit* (druckbar vs. nicht druckbar) solcher Suspensionen zu klassifizieren [8] [9]. Trotz der geringen Datenmenge lieferte das Modell nützliche Vorhersagen, welche Kombinationen von Füllstoffgehalt und Rheologieparametern druckfähig sind, und half so, geeignete Material- und Prozessparameterbereiche einzugrenzen [10] [11]. Klassische Algorithmen sind zudem meist schneller trainiert und leichter interpretierbar. Einschränkend ist, dass ihre Vorhersagegenauigkeit bei sehr komplexen Beziehungen limitiert sein kann, sofern nicht ausreichend Features oder nichtlineare Komponenten berücksichtigt werden.

Probabilistische und bayessche Verfahren: Probabilistische ML-Methoden modellieren Unsicherheiten explizit und eignen sich besonders, wenn Daten knapp sind und eine Quantifizierung der Vorhersageunsicherheit wichtig ist. Ein prominentes Beispiel ist die **Gaussian Process Regression (GPR)**, ein nichtparametrisches bayessches Verfahren, das für kleine Datensätze exzellente Ergebnisse liefern kann [4] [12]. GPR legt eine Gauß-Prozess-Prior-Verteilung über Funktionsräumen und liefert neben der Vorhersage auch eine statistische Vertrauensbandbreite. Dies ist nützlich in unserem Kontext, da das Modell nicht nur eine Eigenschaft (z.B. Härte) für eine gegebene Zusammensetzung schätzt, sondern auch die Sicherheit dieser Schätzung – ein Aspekt, der für experimentelle Planung (aktive Lernverfahren) entscheidend ist. GPR wird in Materialdesign häufig eingesetzt, insbesondere als **Surrogatmodell** im Rahmen von Bayesian Optimization (siehe unten) [13] [14]. Darüber hinaus gehören **Bayessche neuronale Netze** und **Probabilistische Graphische Modelle** (wie Bayessche Netzwerke) zu dieser Kategorie. Ein Bayessches Netzwerk könnte z.B. genutzt werden, um kausale Beziehungen zwischen Zutaten und Eigenschaften zu modellieren (etwa „hoher Photoinitiator-Gehalt erhöht Aushärtetiefe-Wahrscheinlichkeit“) und Unsicherheit durch priorisiertes Expertenwissen zu berücksichtigen. Während Bayessche Netzwerke in der Literatur des Material Machine Learning weniger häufig berichtet sind als GPs, können sie prinzipiell durch Einbindung von Domänenwissen (in Form von Netzwerkstruktur oder Priors) bei sehr kleinen Daten helfen. Generell sind probabilistische Ansätze gut geeignet, um **kleine, verrauschte Datensätze** auszunutzen und dabei die eigene Vertrauenswürdigkeit anzuzeigen [13] [14]. Allerdings skalieren sie oft schlecht mit sehr großen Datensätzen (GPR hat z.B. eine kubische Trainingskomplexität bzgl. der Datenzahl).

Tiefe neuronale Netze (Deep Learning): Deep-Learning-Modelle (z.B. vollverbundene Feedforward-Netze, konvolutionale Netze oder rekurrente Netze) bieten hohe Ausdrucksstärke und können sehr komplexe Muster erlernen – vorausgesetzt, es stehen genügend Daten zur Verfügung. In der Materialinformatik werden *Deep Neural Networks* zunehmend eingesetzt, um Zusammensetzungen oder Molekülstrukturen auf Eigenschaften abzubilden [15] [16]. Ein Vorteil tiefer Netze ist, dass sie **automatisches Feature Learning** betreiben – d.h. sie können hochdimensionale Rohdaten (z.B. molekulare Deskriptoren, Spektraldaten) verarbeiten und abstrakte Merkmale extrahieren, ohne dass der Anwender manuell Feature-Engineering betreiben muss. Für unser Rezepturproblem könnte ein Deep Neural Network z.B. die Anteile aller möglichen Komponenten als Input erhalten (ggf. plus zusätzliche Merkmale jeder Komponente wie Molekülstruktur oder funktionelle Gruppen) und lernt, daraus die Zielgrößen zu prognostizieren. In der Praxis sind tiefe Netze bei sehr knappen Daten jedoch anfällig für Überanpassung. Eine 2023er Übersichtsarbeit über *Small Data ML in Materials Science* betont,

dass herkömmliche Deep-Learning-Modelle typischerweise enorme Datenmengen erfordern und daher in Materialien-Anwendungen oft (noch) unpassend sind, da viele Materialsdatensätze eher klein sind [17] [3]. Dennoch gibt es Strategien, Deep Learning auch bei begrenzten Daten einzusetzen: *Transfer Learning* (Vortrainieren auf großen Datensätzen ähnlicher Aufgaben) und *Multi-Task Learning* (gleichzeitiges Trainieren auf mehreren Eigenschaften oder Datenquellen) können die Effektivität steigern [18] [19]. Ein aktuelles Beispiel liefert Jain *et al.* (2024), die drei Acrylatmonomere in variablen Mischungsverhältnissen untersuchten: Durch einen iterativen aktiven Lernansatz konnten sie mit relativ wenigen Experimenten ein neuronales Modell trainieren, das die mechanischen Eigenschaften (Elastizitätsmodul, Zugfestigkeit, Bruchdehnung, Härte) der Photopolymermischung vorhersagt. Dieses Deep-Learning-Modell wurde anschließend genutzt, um gezielt Zusammensetzungen mit einem gewünschten Elastizitätsmodul zu finden – innerhalb von 6 Iterationen identifizierte das System Formulierungen, die den Zielwert bis auf 10% Genauigkeit erreichten [20] [21]. Ein besonderer Zweig des Deep Learning, der im Kontext von Materialformulierung sehr vielversprechend ist, sind **Graph Neural Networks (GNNs)**. GNNs ermöglichen es, strukturelle Informationen von Molekülen oder komplexen Mixturen als Graphen zu repräsentieren und in die Modellierung einzubeziehen. So kann man z.B. jedes Molekül (Monomer, Oligomer, Additiv) als Graph von Atomen modellieren und dann ein Gesamtmodell aufbauen, das mehrere solcher Molekülgraphen samt ihrer Mengenanteile verarbeitet. Aktuelle Arbeiten zeigen, dass GNN-Modelle in der Polymerinformatik klassische Modelle oft übertreffen und robustere Vorhersagen selbst bei moderaten Daten liefern können [22] [23]. Das *PolymerGNN*-Modell etwa repräsentiert Polymermaterialien als Menge von Monomer-Bausteinen (jeweils als Graph) und erreicht im multitasking Lernen hoher Genauigkeit für verschiedene Eigenschaften (z.B. Glastemperatur und intrinsische Viskosität) – sogar im **low-data regime**, indem chemisches Vorwissen in die Graphdarstellung einfließt [22] [23]. Für unser System könnten GNNs insbesondere dann interessant sein, wenn wir die genaue chemische Struktur der Komponenten berücksichtigen wollen (z.B. Unterschiede zwischen verschiedenen Monomeren oder Photoinitiatoren auf molekularer Ebene). Deep Learning ist insgesamt prädestiniert für **viel Daten** und komplexe Muster; mit speziellen Architekturen und Wissensintegration (siehe unten) können tiefe Netze aber auch aus kleinen Daten Wissenswertes extrahieren.

Symbolische und wissensbasierte Ansätze: Über rein datengetriebene Methoden hinaus gibt es ML-Ansätze, die **explizites Expertenwissen, Logikregeln oder Ontologien** einbeziehen. Diese Ansätze sind besonders relevant, da in unserem Anwendungsfall bereits ein *Ontologie*-Schema der Materialklassen und experimentellen Konzepte vorliegt (siehe Abbildung unten).

Beispiel einer Ontologie-Hierarchie für Materialklassen und verwandte Entitäten. Ontologien definieren Konzepte (z.B. *Monomer*, *Photoinitiator*, *Messung*, *Gerät* etc.) und Relationen zwischen ihnen in formal strukturierter Weise. Dieses Hintergrundwissen kann für ML genutzt werden, um Feature-Engineering zu betreiben oder um Suchräume einzuschränken. Beispielsweise könnten mithilfe der Ontologie kategorische Features generiert werden wie „Anzahl der Komponenten pro Klassenkategorie“ oder „Gesamtvolumenanteil aller Photoinitiatoren“, was dem Modell wichtige Kontextinformationen liefert (etwa dass ein Photoinitiator vorhanden sein muss, damit die Mischung überhaupt UV-härtbar ist). Symbolische *Regelsysteme* und **Induktive Logische Programmierung** (ILP) könnten eingesetzt werden, um aus Daten interpretierbare Regeln abzuleiten (etwa: „Wenn Füllstoffvolumen > X und kein Dispergiermittel, dann *nicht druckbar*“). Allerdings sind ILP-Methoden datenhungrig und komplex; stattdessen populärer sind **symbolisch-numerische Hybridmethoden**, in denen man logisches Wissen mit statistischen Modellen kombiniert. Ein Beispiel dafür sind *Constraints* oder Regularisierung auf Basis von Fachwissen: So könnte man z.B. eine **Nebenbedingung** ins Modell einbauen, dass die Summe aller Komponentenanteile 100% ergeben muss (damit das Modell keine physikalisch unmöglichen Rezepturen vorschlägt), oder eine Strafe im Kostenfunktional, wenn bekannte Unverträglichkeiten kombiniert werden. In neueren Forschungsarbeiten werden **Knowledge Graph Embeddings** eingesetzt, um Ontologie-Wissen numerisch zu repräsentieren: dabei wird die Ontologie als Graph aus

Entitäten und Relationen betrachtet, und mittels Techniken wie *OWL2Vec* werden für jedes Konzept kontinuierliche Vektoren gelernt [24] [25]. Solche Vektoren können dann als zusätzliche Eingaben ins ML-Modell einfließen. Huang *et al.* (2024) zeigen z.B., dass die Integration eines Wissensgraphen über chemische Element-Eigenschaften die Vorhersagegenauigkeit für Materialeigenschaften signifikant steigern kann [26] [27]. In ihrem Ansatz (genannt **ESNet**) wurde ein vorab trainierter *Element-Knowledge-Graph Encoder* mit einem kristallographischen Graphmodell fusioniert, um die Materialrepräsentation zu verbessern – dadurch sank der Vorhersagefehler für z.B. Bandlücken spürbar im Vergleich zu einem Modell ohne Wissensgraph [28] [29]. Dieses Prinzip ließe sich analog auf unsere Polymer-Komposit-Domäne übertragen, indem z.B. eine Ontologie der Komponenten (inklusive chemischer Attribute wie Funktionalitäten, molare Massen etc.) in Vektoren codiert und dem Modell als zusätzliches Wissen dient. Ein weiterer symbolischer Ansatz ist **symbolische Regression** (z.B. mit genetischer Programmierung), welche versucht, eine geschlossene analytische Formel herzuleiten, die die Daten beschreibt [30] [31]. In Materials Science wurde symbolische Regression bereits verwendet, um interpretable **Deskriptoren** zu finden – z.B. entdeckten Weng *et al.* (2019) mit genetischer Programmierung eine einfache Kombination von physikalischen Materialparametern, die die Katalysatorleistung gut vorhersagt [32] [33]. Für Resin-Formulierungen könnte symbolische Regression theoretisch genutzt werden, um empirische Formeln (vielleicht angelehnt an Modelle der Rheologie oder Optik) zu identifizieren, die Aushärtetiefe oder Viskosität in Abhängigkeit der Inhaltsstoffe beschreiben. Vorteil solcher symbolischer Ausdrücke ist ihre **Erklärbarkeit** und die Möglichkeit, sie mit bekannten wissenschaftlichen Gesetzen zu vergleichen. Allerdings stoßen rein symbolische Verfahren bei hochdimensionalen Mischungssystemen schnell an Grenzen, sodass sie vermutlich eher als Ergänzung (zur Hypothesengenerierung) denn als alleiniges Werkzeug dienen.

Zusammenfassend lässt sich festhalten, dass es eine breite Palette an ML-Methoden gibt, die für die **Eigenschaftsvorhersage** und **Formulierungsoptimierung** von 3D-Druck-Polymerkompositen in Frage kommen. Welche Methode geeignet ist, hängt maßgeblich von den verfügbaren **Datenmengen** und dem **Vorwissen** ab. Nachfolgend betrachten wir diese Aspekte genauer und beleuchten auch, welche Verfahren sich speziell für Vorhersage oder Optimierung eignen.

Umgang mit Datenmengen: Wenig vs. viel Daten

In vielen Bereichen der Materialinformatik – speziell bei neuen Materialsystemen – ist die Datenlage zu Beginn begrenzt (*„small data“* Problem [17] [34]). Unsere Anwendung bildet hier keine Ausnahme: Historische Rezepturdaten liegen nur spärlich vor, weshalb ein geeignetes ML-System mit **wenigen, teilweise unvollständigen Datenpunkte** auskommen muss. Gleichzeitig besteht die Option, über Simulation oder systematische Experimente zusätzliche Daten zu generieren, sodass perspektivisch auch ein **großer Datensatz** verfügbar sein könnte. Je nach Datenregime ändern sich die favorisierten Methoden und Strategien:

- **Methoden für kleine Datenmengen:** Bei wenigen Datenpunkten (z.B. < 100 oder ein paar hundert Samples) ist das Vermeiden von Overfitting und die Nutzung von **Zusatzwissen** entscheidend. Wie oben diskutiert, haben bestimmte Algorithmen erwiesenermaßen eine gute *small-data*-Verträglichkeit: dazu zählen SVM, Gaussian Processes, Entscheidungsbaum-Ensembles (Random Forest, Gradient Boosting wie XGBoost) und auch symbolische Regression [3] [4]. Diese Modelle haben eingebauten Regularisierungsmechanismen oder eine begrenzte Komplexität, sodass sie nicht so leicht jedem Rauschen im Datensatz folgen. Eine aktuelle Review bestätigt explizit, dass gerade SVM, GPR und Boosting-Bäume in Materials-Anwendungen oft mit kleinen Datensätzen ideale Leistungen erzielen [3] [4]. Ergänzend kommen auf *Strategie-Ebene* Techniken wie **Transfer Learning** und **Active Learning** zum Tragen [13] [14]. Transfer Learning bedeutet, ein Modell auf verwandten, größeren Datensätzen vorzutrainieren – etwa könnte man neuronale Netze zunächst auf allgemeineren Polymerdaten (aus öffentlichen Datenbanken)

trainieren und dann mit den wenigen verfügbaren Resin-Daten feinjustieren [19] [35]. Active Learning wird unten im Abschnitt Optimierung detaillierter beschrieben, aber im Kern wählt hierbei das Modell *selbst* die informativsten neuen Datenpunkte aus, um zielgerichtet den kleinen Datensatz zu erweitern [13] [14]. Weitere wichtige Ansätze für kleine Daten sind **Datenaugmentation** (z.B. synthetische Daten generieren, indem man bekannte physikalische Gesetzmäßigkeiten nutzt oder Zufallsrauschen hinzufügt) und striktes Cross-Validation/ Evaluationsprotokoll, um Überanpassung rechtzeitig zu erkennen. Domain-Wissen kann auch helfen, die **Feature-Dimension** zu reduzieren: Wenn die Eingabe-Feature-Zahl groß ist relativ zur Datenzahl, steigt das Overfitting-Risiko (Hughes'ches Phänomen). Durch Feature Selection oder Bildung aussagekräftiger Summen-Features (z.B. berechnete Kenngrößen wie *Oberflächenenergie der Monomer-Mischung*, *Viskositätsindex* etc.) kann man den effektiven Suchraum eindampfen. Literatur zeigt, dass das Generieren von Deskriptoren auf Basis von Expertenwissen die Modellleistung in small-data Fällen signifikant verbessern kann [36] [37]. Beispielsweise konstruierte Zhao *et al.* (2023) in einer Alloy-Studie 309 initiale Features aus Element-Eigenschaften und reduzierte diese via Korrelations- und Wichtigkeitsanalyse auf 5 Schlüsselfeatures, um daraus mit nur 32 Datenpunkten ein präzises Modell für die Härtevorhersage zu bauen [38] [39].

- **Methoden für große Datenmengen:** Falls durch Hochdurchsatz-Experimente oder Simulation tausende bis zehntausende Datenpunkte zur Verfügung stehen, können wir die Leistungsfähigkeit komplexerer Modelle voll ausschöpfen. Tiefes Lernen wird dann attraktiv – größere Datensätze erlauben den Train von **mehrschichtigen neuronalen Netzen** oder sogar die Erprobung von **Deep Learning Architektur-Suche**. Mit Big Data kann man auch vermehrt **unüberwachte Methoden** einsetzen, um Muster zu entdecken (z.B. Clustering von Formulierungen nach ähnlichem Verhalten) oder **selbstüberwachte Vortrainings** (z.B. Autoencoder, die latente Darstellungen von Rezepturen lernen). Dennoch sollte man beachten, dass in der Materialwissenschaft selbst 1000 Datenpunkte oft schon als viel gelten, während in typischen Deep-Learning-Anwendungen (Bilderkennung, NLP) Millionen von Daten vorliegen. Daher sind in vielen Fällen Hybridansätze aus datenhungrigen und datenfreundlichen Methoden sinnvoll. Sollte wirklich ein sehr umfangreicher Datensatz vorliegen (etwa durch Simulation generiert), kann auch das Training von **Ensemble-Models** lohnend sein – z.B. das Trainieren vieler unterschiedlicher Architekturen und das Mitteln ihrer Vorhersagen. Gu *et al.* (2023) berichten beispielsweise, dass ein Ensemble aus verschiedenen Machine-Learning-Modellen (darunter Boosting und neuronale Netze) genutzt wurde, um gleichzeitig Härte, Zähigkeit und Festigkeit von 3D-Druck-Photopolymeren hochpräzise vorherzusagen [40]. Große Datenmengen erlauben ferner die Aufsplittung in separate Trainings- und Validierungs/Testsätze, sodass hyperparameter tuning und Modellvergleich unter realistischen Bedingungen möglich wird.

Abschließend sei betont, dass die Grenze zwischen *„wenig"* und *„viel"* Daten fließend ist [34] [41]. Oft wird man im Materialdesign zunächst mit wenigen Daten starten und mittels aktiver Lernverfahren oder zusätzlichen Simulationen den Datensatz iterativ vergrößern – das ML-Modell muss sich also an unterschiedliche Datenregime anpassen können. In frühen Phasen dominieren wissensgestützte und robuste Verfahren, während mit wachsender Datenmenge komplexere Modelle ihre Stärken ausspielen können. Es gibt **kein universelles Modell**, das für alle Datenumfänge optimal ist [42] [43]; vielmehr sollte die Methodenauswahl dynamisch an den verfügbaren Datenbestand angepasst werden.

Vorhersageaufgabe vs. Optimierungsaufgabe

Wie eingangs erläutert, lassen sich zwei Hauptaufgaben unterscheiden: **Eigenschaftsvorhersage** (forward prediction) und **Materialoptimierung** bzw. Inverse Design. Beide erfordern unterschiedliche methodische Schwerpunkte, die wir hier beleuchten.

Vorhersage (Forward Modeling): Hierbei wird ein Modell trainiert, das Eingaben (x) (eine spezifische Rezeptur mit bestimmten Komponenten und Konzentrationen) auf Zielgrößen (y) (eine oder mehrere Eigenschaften oder Qualitätsmetriken) abbildet: $y = f(x)$. Typischerweise handelt es sich um überwachte Lernprobleme – *Regression*, wenn y kontinuierlich (z.B. Härtewert in Shore A) ist, oder *Klassifikation*, wenn y kategorisch/binär (z.B. *druckbar* ja/nein) ist. Für die Vorhersage sollte das Modell möglichst generalisierungsstark und präzise sein. Modellgüte wird mit üblichen Metriken bewertet: z.B. **RMSE** (Root Mean Square Error), **MAE** (Mean Absolute Error) oder **R^2** für Regression; **Genauigkeit**, **Precision/Recall**, **F1-Score** oder **AUROC** für Klassifikation. Bei multiplen Ausgabegrößen (Multi-Output Regression) können Fehlermetriken für jede Eigenschaft separat betrachtet werden oder man verwendet zusammengefasste Kennzahlen (z.B. gewichtete Fehlersumme). In der Praxis hat es sich bewährt, **Ensemble-Ansätze** oder **Multi-Task-Learning** einzusetzen, wenn mehrere Zielgrößen parallel vorhergesagt werden sollen [44] [22]. Multi-Task-Netzwerke teilen dabei zunächst gemeinsame Lernrepräsentationen und verzweigen sich dann in aufgabenspezifische Ausgaben – das kann die Daten effektiv „teilen“ und verhindert, dass für jede Zielgröße ein eigenes großes Modell gelernt werden muss. In unserer Domäne könnte man etwa Härte, Zugfestigkeit und Aushärtetiefe gemeinsam vorhersagen lassen, da sie sich alle aus der Mikrostruktur der gehärteten Harz/Keramik-Matrix ergeben und somit korrelierte Grundlagen haben. Ein guter Validierungsvorgehen für Vorhersagemodelle mit wenig Daten ist strenges **Cross-Validation** (z.B. Leave-One-Out oder k-Fold), um sicherzustellen, dass das Modell nicht nur die bekannten Rezepturen auswendig lernt, sondern wirklich unbekannte Kombinationen richtig einschätzt. Interpretierbarkeit ist bei Vorhersagemodellen oft ein Thema: Während Black-Box-Modelle wie tiefe Netze hohe Genauigkeit bringen, bieten z.B. Entscheidungsbäume oder symbolische Regression direkt einsichtige Entscheidungsregeln oder Formeln. In industriellen Kontexten kann ein **handhabbarer Kompromiss** darin bestehen, ein hochgenaues komplexes Modell zu nutzen, aber ergänzend **SHAP-Werte** oder andere Feature-Importance-Techniken zur Erklärung beizuziehen, welche Eingaben die Prognose dominieren.

Optimierung (Inverse Design): Die inverse Fragestellung – „finde x, sodass Eigenschaft y möglichst optimal (z.B. maximal, minimal oder nahe einem Zielwert) wird“ – erfordert ein systematisches Durchsuchen des Input-Raums. Da dieser Raum (alle möglichen Kombinationen der Materialkomponenten und Konzentrationen) bei typischen Rezepturen gewaltig ist, kommen heuristische oder modellbasierte Optimierungsverfahren zum Einsatz. Wichtig ist hier zu unterscheiden, ob man realweltliche Experimente durchführt (d.h. jede getestete Rezeptur ist teuer und Zeitaufwändig) oder ob man auf einem bereits trainierten Surrogatmodell operiert. Im *Experiment-Loop* ist **Bayesian Optimization (BO)** derzeit State-of-the-Art für Materialoptimierung [13] [45]. BO kombiniert ein probabilistisches Surrogat (oft GPR, kann aber auch Random Forest oder Bayessches NN sein) mit einer **Erwerbsfunktion** (acquisition function), welche darüber entscheidet, welcher Versuchspunkt als nächstes ausprobiert werden sollte. Die Erwerbsfunktion (z.B. *Expected Improvement*, *Upper Confidence Bound* oder *Thompson Sampling*) balanciert **Exploitation** (Ausnutzen des aktuellen Modells, um im vermuteten Optimum zu suchen) und **Exploration** (gezielt Bereiche mit hoher Unsicherheit erkunden) [45] [46]. In frühen Iterationen, wenn das Modell noch unsicher ist, wählt BO tendenziell diverse, informative Punkte; mit mehr Daten shiftet die Strategie zu exploitation, um das Optimum feinzujustieren [47] [48]. BO eignet sich hervorragend für kleine Datenprobleme, da es mit jedem neuen Experiment dazulernt und das Maximum mit **wenigen Versuchen** finden kann. Ein Beispiel aus der Photopolymer-Optimierung liefert Jo *et al.* (2022): Sie nutzten **Multi-Objective Bayesian Optimization** mit GPR, um simultan die Schichtdicke, Photoinitiatoranteil und Füllstoffgehalt so einzustellen, dass mechano-lumineszente Harze maximal leuchten, aber die Druckzeit minimal bleibt [49] [50]. Das GPR-Modell lieferte Pareto-optimale Lösungen, die dann experimentell validiert wurden. In nur wenigen Iterationen konnten so deutlich verbesserte Rezepturen identifiziert werden, die einen guten Kompromiss zwischen den konkurrierenden Zielen darstellten. Ähnlich beeindruckend ist die Arbeit von Erps *et al.* (2021), die mit einem automatisierten BO-Framework neue 3D-Druck-Formulierungen mit **multioptimalen mechanischen Eigenschaften** entdeckten: Bereits nach 30 experimentellen

Mischungen fanden sie 12 optimale Rezepturen und erweiterten den erreichbaren Eigenschaftsraum um das 288-fache im Vergleich zum Ausgangsmaterial [51] [52]. Dies zeigt das Potenzial von ML-geführter Optimierung, innerhalb kurzer Zeiträume Materialien zu finden, die man mit manuellem Probieren kaum gefunden hätte.

Wenn man bereits ein gut validiertes Vorhersagemodell $f(x)$ besitzt, kann man die Optimierung auch **in silico** darauf durchführen. Ein einfacher Ansatz ist **Grid Search** oder randomisiertes Sampling: dabei generiert man viele virtuelle Kandidaten (ggf. unter Verwendung der Ontologie, um sinnlose Kombinationen zu vermeiden) und lässt das Modell deren y prognostizieren, um die besten auszuwählen. Bei hochdimensionalen Mischungen ist das aber u.U. ineffizient. Klügere Verfahren sind **evolutionäre Algorithmen** wie der *Genetische Algorithmus* (GA) oder *Particle Swarm Optimization* (PSO), die die Rezeptur als „Chromosom" bzw. Partikel behandeln und durch Evolution bzw. Schwarmverhalten optimieren. Solche heuristischen Methoden wurden im Materialdesign bereits benutzt – z.B. wurde berichtet, dass GA und PSO erfolgreich zur Optimierung von ML-Modellhyperparametern für Polymerblend-Härtevorhersagen eingesetzt wurden [53] [54], was nahelegt, dass sie auch direkt auf die Materialkombination anwendbar sind. Der Vorteil von GA/PSO ist, dass sie **kein Surrogat mit Unsicherheitsmaß** brauchen und auch nicht-differenzierbare Zielvorgaben (oder mehrere Ziele) handhaben können, etwa durch Definieren einer Fitness-Funktion. Allerdings verlangen sie typischerweise **viele Evaluierungen** – wenn jedes Evaluation ein reales Experiment wäre, ist das teuer; falls man aber zunächst auf dem ML-Modell sucht, kann man relativ gefahrlos tausende Iterationen simulieren und erst die vielversprechendsten Lösungen real testen. Ebenso können **reinforcement learning** (RL) Ansätze formuliert werden, in denen ein Agent schrittweise Zutaten hinzufügt und dafür einen Belohnungsscore (basierend auf der Modellausgabe) erhält. RL ist jedoch bei strukturierter Ausgabe wie einer Zutatenliste anspruchsvoll und in Materials-Optimierung noch selten.

Besonders zu beachten sind **Mehrziel-Optimierungen**: In unserem Fall möchte man vielleicht eine Rezeptur, die *gleichzeitig* z.B. maximale Härte und minimale Schrumpfung erzielt. Solche Ziele konkurrieren oft, sodass es keine einzelne optimale Lösung gibt, sondern eine Menge von **Pareto-optimalen** Lösungen (Trade-offs). Multiobjektive Evolutionäre Algorithmen (wie NSGA-II) können eine Population entlang der Pareto-Front konvergieren lassen. Multiobjektive BO-Methoden verwenden *acquisition functions*, die die Verbesserung in mehreren Zielen bewerten (z.B. *Expected Hypervolume Improvement*). Jo *et al.*'s oben genanntes Beispiel war eine solche multi-kriterielle Optimierung mittels BO [49] [55]. Es zeigte sich, dass BO effizient geeignete Kompromisse finden konnte, was in praktischen Anwendungen bedeutet: Der Entwickler kann aus mehreren vorgeschlagenen Rezepturen diejenige auswählen, die seinen Präferenzen am besten entspricht (z.B. „etwas weniger hart, dafür deutlich glattere Oberfläche").

In allen Optimierungsszenarien kann die bereits erwähnte **Active-Learning-Schleife** zum Einsatz kommen. Dabei verschwimmen Vorhersage und Optimierung: Man nutzt das aktuelle Modell, um die vielversprechendste nächste Experimentier-Kombination vorzuschlagen (Optimierungsschritt), führt das Experiment durch und ergänzt die Daten (Vorhersagemodell wird retrainiert) – und iteriert weiter. Diese *Closed-loop* Strategie wurde jüngst in verschiedenen Materialsystemen demonstriert. Ein Beispiel aus unserem Themenbereich: Ayush Jain *et al.* (2024) kombinierten aktives Lernen mit einem ML-Vorhersagemodell, um aus einem ternären Monomerraum gezielt neue Photopolymerformulierungen auszuwählen. Durch adaptives Sampling konnten sie mit minimalen Experimenten ein Modell aufbauen, das Young'sches Modul, Härte usw. präzise vorhersagt, und anschließend das Modell verwenden, um gezielt eine Zusammensetzung mit gewünschtem Modul zu designen [20] [56]. Die Erfolgsquote war hoch – der vorgeschlagene Kandidat lag innerhalb von 10% des Ziel-Steifigkeitswertes [57]. Auch in der Metalllegierungs-Entwicklung wurde Active Learning eingesetzt, z.B. suchten Xue *et al.* (2021) mithilfe eines SVR-Modells und **Efficient Global Optimization (EGO)** nach neuen Formulierungen von Formgedächtnislegierungen mit niedriger Hysterese. Über 9 Iterationen wurde der

Datensatz schrittweise erweitert; am Ende wiesen 14 der 36 getesteten neuen Legierungen bessere Eigenschaften auf als alle ursprünglichen 22 – ein deutlicher Erfolg [58] [59]. Solche iterativen Erfolge sind ein starkes Indiz dafür, dass ML in der Lage ist, in Materialsystemen *überraschende* Neuentdeckungen zu machen, indem es nicht-intuitive Kombinationen findet, die dennoch funktionieren.

Zusammengefasst gilt: Für **Vorhersagen** wählt man Modelle, die akkurat und verallgemeinerungsfähig sind; für **Optimierung** benötigt man Strategien, die den Rezepturraum intelligent durchsuchen und dabei die Datenökonomie beachten. Oft gehen beide Hand in Hand: Ein gutes Vorhersagemodell ist die Grundlage für Simulation oder Bayesian Optimization, und umgekehrt liefert die Optimierung neue Datenpunkte zur Verbesserung des Modells.

Einbindung von Expertenwissen: Ontologien und hybride KI

Ein Alleinstellungsmerkmal unseres Szenarios ist die vorhandene **Ontologie** bzw. das explizite **Fachwissen** über Materialklassen, Prozesse und Constraints. Die Frage stellt sich, *welche ML-Methoden dieses Wissen aktiv nutzen können*, um bessere Ergebnisse zu erzielen.

Ontologie- oder wissensgestützte ML ist ein aktives Forschungsfeld. Einige Möglichkeiten zur Integration sind: (a) **Feature Engineering mit Ontologie** – d.h. neue Eingabegrößen aus Ontologiekonzepten ableiten; (b) **Wissensgraph-Embeddings** – das Ontologienetzwerk in Vektoren abbilden und ins Modell einspeisen; (c) **Constraints und Regularisierung** – Fachwissen als Regeln in den Lernalgorithmus einbauen; (d) **Wissensbasierte Inferenzsysteme** – separat vom ML-Modell logische Schlussfolgerungen ziehen und deren Output dann kombinieren.

Manche ML-Algorithmen sind besonders geeignet, Wissen zu verarbeiten. **Graphbasierte Methoden** (wie Graph Neural Networks oder Graphenkernmethoden) können Ontologie-Graphen direkt aufnehmen. Ein Ansatz wäre z.B., einen großen Knowledge Graph zu bauen, der Materialien, ihre Klassen, Eigenschaften und eventuelle experimentelle Beobachtungen verknüpft. Algorithmen wie Node2Vec oder OWL2Vec können aus solchen Graphen **Embeddings** lernen, die latente Beziehungen zwischen Konzepten codieren [24] [25]. Diese Embeddings können dann an ein Prädiktionsmodell übergeben werden. So könnte man jedem Materialbestandteil (z.B. einem konkreten Monomer) einen Ontologie-basierten Vektor zuordnen, der z.B. enthält, dass es zur Klasse *Acrylat* gehört, eine bestimmte Molmasse hat, etc. Das ML-Modell bekommt dann pro Komponente diesen Vektor anstatt (oder zusätzlich zu) einem simplen Kategorien-Index. Dadurch wird semantische Ähnlichkeit berücksichtigt: Zwei ähnliche Monomere erhalten ähnliche Vektoren, was dem Modell ermöglicht, generalisierte Schlüsse zu ziehen, selbst wenn ein bestimmtes Monomer in den Trainingsdaten selten vorkam – es „weiß“ aus der Ontologie, dass es z.B. einem bekannten Monomer ähnlich ist. In der zitierten Arbeit von Fang *et al.* (2022) etwa wurde für chemische Elemente eine solche Wissensgraph-Einbettung genutzt, um Materialeigenschaftsprognosen zu verbessern [28] [60].

Auch **Regelbasierte ML**-Techniken wie *Inductive Logic Programming* können Ontologien verwerten, indem sie innerhalb der Ontologie nach Mustern suchen, die erfolgreiche vs. fehlgeschlagene Rezepturen trennen. Zum Beispiel könnte ILP aus einigen Beispielen eine Regel „`Formulierung(X) :- enthaelt(X,AdditivY), Klasse(Y,Dispergiermittel), enthaelt(X,Z), Klasse(Z,Keramik), fehlt(X,Klasse(UV-Absorber)).`“ generieren, was in Prolog-artiger Syntax bedeuten könnte: *Eine Formulierung ist erfolgreich druckbar, wenn sie ein Dispergiermittel und Keramikpartikel enthält und kein UV-Absorber fehlt*. Solche Regeln wären direkt aus der Ontologie ableitbar und für Experten nachvollziehbar. Allerdings erfordert ILP meist mehr Daten als

wir haben und kann in realen Mischungen, wo viele Komponenten gemeinsam wirken, sehr komplex werden.

Eine praktischere Herangehensweise ist das Definieren von **Constraints** bei der Optimierung. Mithilfe der Ontologie könnten wir z.B. die *Suchraumgrenzen* einschränken: Wenn bekannt ist, dass bestimmte Komponenten nicht zusammen stabil sind (z.B. bestimmte Photoinitiator-UV-Absorber-Kombinationen, die sich gegenseitig deaktivieren), kann man diese Kombinationen in der Kandidatenauswahl ausschließen. Ebenso kann man Schwellwerte aus der Literatur als *harte Nebenbedingungen* implementieren – etwa „Viskosität < X Pa·s für Druckbarkeit" – um unbrauchbare Rezepturen gar nicht erst vorzuschlagen. Solche Constraints können in Evolutionären Algorithmen und BO integriert werden (BO kann z.B. über das acquisition function die penalisierten Bereiche meiden). Ein Beispiel: Nasrin *et al.* (2024) nutzten ihr ANN-Modell zur Druckbarkeitsklassifikation, um einen *Map* zu erstellen, welche Kombinationen von Partikelgrößenverteilung und Viskosität druckbar sind [10] [11]. Diese Informationen entsprechen letztlich logischen Regeln (z.B. *bimodale Verteilung & Viskosität Y -> druckbar*), die man bei der Rezepturplanung berücksichtigen kann.

Manche ML-Frameworks erlauben auch direkt, **Logik in neuronalen Netzen** abzubilden, z.B. durch *Logic Tensor Networks* oder *DeepProbLog*, wo logische Ausdrücke in Differenzierbarkeit überführt werden. Das ist allerdings noch sehr experimentell und kompliziert.

Praktisch gesehen, ist eine sinnvolle Ontologie-Integration oft im **Preprocessing** und **Feature-Design** verortet: Die Ontologie stellt sicher, dass alle Datenpunkte einheitlich klassifiziert sind (z.B. gleiche Benennung und Gruppierung der Inhaltsstoffe, Entitäten für Messmethoden usw.), was Datenbereinigung und -zusammenführung erleichtert [61]. Außerdem kann man Ontologie nutzen, um **Datenlücken zu füllen** – wenn z.B. in alten Protokollen nur ein Handelsname eines Monomers stand, kann man über das Ontologie-Wissensnetz herausfinden, um welche chemische Substanz es sich handelt und ihre Eigenschaften nachtragen. So verbessert das Wissen indirekt die Datenqualität [62] [63].

Insgesamt können Ontologien und explizites Wissen die ML-Pipeline an vielen Stellen stützen: Von konsistenter Datencodierung über intelligentere Features bis hin zu besserer Interpretierbarkeit der Modelle. Wichtig ist, Algorithmen zu wählen, die diese Extrainformation verarbeiten können – z.B. GNNs für graphische Wissensrepräsentation oder Bayesian-Optimization, die mit prior beliefs (durch Knowledge-driven priors) starten kann [13] [64]. Einige Studien sprechen von *„Hybrid Intelligence"* oder *„Physics- and Knowledge-informed AI"* in Materialwissenschaft, wo domänenspezifische theoretische Modelle mit ML kombiniert werden [65] [66]. Ein Beispiel wäre die **Beer-Lambert-Gesetz**-Approximation für Aushärtetiefe: man könnte dieses physikalische Gesetz als Feature oder als Teil eines Modell-Ansatzes integrieren (z.B. dem Netzwerk beibringen, exponentielle Intensitätsabnahme mit UV-Absorber-Gehalt nachzubilden). Solche hybriden Modelle (auch *Graue Kästen* genannt) nutzen bekannte physikalische Beziehungen als Grundgerüst und lassen ML nur die unbekannten Restzusammenhänge lernen – was Datenbedarf senkt und Vertrauen erhöht.

Für unser Projekt ist insbesondere ein hybrider Dreiklang interessant: **Ontologie + GNN + Bayesian Optimization** (wie vom Fragesteller angedeutet). Wissenschaftlich erscheint dieser Ansatz sehr schlüssig: Die Ontologie kann als formales Wissensgerüst dienen, GNNs können sowohl den Ontologiegraph als auch chemische Strukturgraphen der Komponenten verarbeiten, und Bayesian Optimization kann auf dem so informierten Modell iterativ optimale Rezepturen vorschlagen. Ein derartiges System würde die Stärke aller drei Welten vereinen: erklärbares Expertenwissen, leistungsfähige datengetriebene Vorhersage und effiziente experimentelle Suche. Der **Praxiswert** ist hoch, denn jedes Teilsystem ist bereits in einfacher Form erprobt (Ontologien in Datenmanagement, GNNs in Stoffdateneigenschafts-Prognose, BO in Materialoptimierung). Die Herausforderung liegt in

der technischen Integration – z.B. muss definiert werden, wie genau die Ontologie ins GNN eingeht (etwa als zusätzlicher Graph, der in eine gemeinsame latente Repräsentation gemappt wird). Es gibt aber erste Arbeiten in ähnlich gelagerten Bereichen, die Machbarkeit demonstrieren [26] [27]. Wichtig ist, die **Komplexität im Auge zu behalten**: Ein allzu kompliziertes hybrides System könnte schwer zu trainieren oder abzustimmen sein, vor allem wenn Daten knapp sind. Daher wäre ein gestufter Ansatz ratsam: zunächst Ontologie nutzen, um klassische Modelle zu verbessern (Feature-Engineering, Datenaufbereitung), dann mit mehr Daten GNNs einführen, und schließlich, wenn das Vorhersagemodell stabil ist, die BO-Optimierung in den Loop nehmen.

Vorverarbeitung, Architekturentscheidungen und Metriken

Damit ML-Modelle im beschriebenen Setting erfolgreich sind, müssen einige praktische Aspekte beachtet werden: **Datenvorbereitung**, geeignete **Modellarchitekturen** für die Eingaberepräsentation, ausreichende **Datenmengen** pro Modellkomplexität sowie sinnvolle **Evaluierungsmethoden**.

Datenvorverarbeitung: Die historischen Daten sind „teils unsauber" – das bedeutet vermutlich, dass Einträge fehlen, unterschiedliche Einheiten/Skalen genutzt wurden, oder Messwerte Rauschen enthalten. Eine gründliche **Datenbereinigung** ist der erste Schritt: Konsistente Einheiten (z.B. alle Konzentrationen in Gewichts- oder Volumenprozent), Korrektur offensichtlicher Tippfehler, Behandlung von fehlenden Werten (Imputation mit physikalisch sinnvollen Annahmen oder simply Weglassen solcher Samples, je nach Ausmaß). Gegeben der Ontologie sollten alle Zutaten eindeutig klassifiziert sein – z.B. sollte klar sein, welcher Stoff als *Photoinitiator* zählt, etc. Hier hilft es, Aliasnamen über Ontologieverknüpfungen zu vereinen. **Feature-Skalierung** ist wichtig für ML-Modelle: Konzentrationswerte und Messwerte sollten normalisiert werden (z.B. 0–1 Skalierung oder Standardisierung), damit kein Feature das Modell dominiert nur aufgrund seines Zahlenbereichs. Für einige Eingabegrößen bietet sich eine **log-Transformation** an – z.B. Viskositäten oder Aushärtetiefen, die oft über mehrere Größenordnungen schwanken, könnte man log10 transformieren, um extreme Schiefe aus den Daten zu nehmen.

Ein Spezialthema ist die Repräsentation der **Rezeptur-Komposition** als Input für ML. Da eine Formulierung variabel viele Komponenten enthalten kann (und es theoretisch hunderte von möglichen chemischen Substanzen gibt), muss man den Inputraum geschickt kodieren. Ein simpler Ansatz ist ein fester Vektor mit Dimension = Anzahl aller möglichen Komponentenarten, der die Anteile enthält (nicht enthaltene Komponenten = 0). Dieser „One-hot concentration vector" funktioniert, wenn die Materialvielfalt überschaubar ist. In unserem Fall könnte man die Komponenten zumindest nach Hauptklassen aufsplitten: z.B. 10 mögliche Monomere, 5 Oligomere, 5 Photoinitiatoren etc., dann ein Feature-Array mit Summe 20-30 Dimensionen, was jeweils die Gehalte dieser konkreten Substanzen enthält. Das setzt voraus, dass man die potenziellen Kandidaten im Voraus festlegt. Alternativ kann man eine **per-ingredient Liste** verwenden und in einem neuronalen Netz z.B. mittels eines *Set2Set*-Modells verarbeiten: Jedes Zutaten-Item wird durch (Klasse, Substanz-ID, Konzentration) beschrieben, und das Modell aggregiert über die Menge, unabhängig von der Reihenfolge. Graph-Neural-Network-Ansätze würden vermutlich so etwas implementieren: Jeder Knoten ist eine Ingredienz, dessen Feature z.B. ein Molekulgraph-Embedding + Konzentration ist; Kanten könnten Interaktionen bedeuten (ggf. voll verbundener Graph zwischen allen Zutaten, oder Zusatzknoten für die Gesamtformulierung). Das ist komplex, aber könnte Synergien im Netzwerk lernen (z.B. dass Photoinitiator und UV-Absorber gegensätzliche Effekte auf Aushärtetiefe haben, etc.). Für tabellarische Modelle ist es oft sinnvoll, zusätzliche zusammengesetzte Features zu geben: z.B. **Verhältnis** bestimmter Komponenten (Monomer/Oligomer-Verhältnis, Füllstoff/Harz-Verhältnis), **Summenwerte** (Summe aller Füllstoffe, Summe aller Additive), oder physikalisch motivierte Deskriptoren (berechnete Gesamt-Viskosität nach Mischungsregel, etc.). Solche abgeleiteten Features leiten Domainwissen ins Modell.

Architekturentscheidungen: Bei klassischen ML-Modellen stellt sich die Frage nicht so sehr, da diese eher algorithmen- als architekturbasiert sind (außer vielleicht Baumtiefe oder Ensemblegröße). Bei neuronalen Netzen jedoch muss man die Netzarchitektur und Layergrößen an Problem und Daten anpassen. Für kleine Daten gilt: ein **kleines Netzwerk** (wenige Schichten, wenige Neuronen) ist oft besser, da es weniger Parameter hat, die überfittet werden könnten. Eine Daumenregel ist, nicht mehr Freiheitsgrade als etwa 5-10x die Anzahl der Datenpunkte zu nehmen. Andernfalls muss man stark regulärisieren (Dropout, Weight Decay, etc.). Wenn man tiefer gehen will, sollte man wie erwähnt Transfer Learning erwägen – z.B. vorab ein ähnliches Problem lernen lassen. In unserer Rezeptur-Vorhersage könnte man z.B. ein neuronales Netz auf einer großen offenen Datenbank polymerer Materialeigenschaften vortrainieren (z.B. PolyInfo oder ähnliches), damit es Grundkenntnisse über Zusammenhänge lernt, und dann fein-tunen. Eine andere architekturelle Frage ist die **Input- und Output-Kodierung**: Wenn Ausgaben qualitativ unterschiedliche Größen sind (z.B. mechanische Festigkeit vs. optische Eigenschaften), könnte man getrennte Ausgabemodule nutzen. Für die Input-Seite könnte man ein **Multi-Branch Network** designen: z.B. ein Ast verarbeitet alle Harzkomponenten, ein anderer Ast alle Füllstoff-bezogenen Eingaben, ein dritter generelle Additiv-Informationen. Diese Äste lernen zunächst intern bestimmte repräsentationen (z.B. der Harz-Ast lernt, wie Monomer/Oligomer/Photoinitiator zusammenwirken für Reaktivität), und am Ende werden alle zusammengeführt (concatenation) und weiterverarbeitet zu den Ausgaben. Solche modularen Architekturen können die Komplexität des Lernens reduzieren, da das Netz bestimmte Teilprobleme isoliert lernt. Graph Neural Networks würden wiederum anders aufgebaut: man müsste definieren, welche Graphstruktur das Netz sieht – z.B. die Molekülstruktur der Komponenten oder einen Interaktionsgraph der Komponentenklassen. Einige aktuelle GNN-Frameworks für Polymere erlauben es, mehrere Monomereinheiten als separate Graphen einzugeben, die dann via **graph pooling** zu einem Gesamtfingerabdruck zusammengeführt werden [67] [68]. Ähnliches wäre für unsere Mischungen denkbar.

Datenmenge vs. Komplexität: Hier nochmals konkret: Sollte nur z.B. $N=50$ Datensätze vorliegen, dann wären vermutlich lineare oder einfache nichtlineare Modelle (GP, SVM mit simplem Kernel, kleiner Random Forest) die sicherste Wahl – alles andere birgt hohes Overfitting-Risiko. Bei $N \approx 200-500$ könnte man einfache neuronale Netze oder mehrstufige Ensemble-Methoden einsetzen, muss aber stark validieren (z.B. wiederholt trainieren mit unterschiedlichen Folds). In diesem Bereich sind auch **Unsicherheitsabschätzungen** wertvoll: Methoden wie GPR oder bayessche NN liefern Konfidenzen, die signalisieren können, wann dem Modell die Extrapolation zu unsicher ist. Ab $N > 1000$ kann man mehr wagen: komplexere Architekturen, intensives Hyperparameter-Tuning oder vollautomatisierte **AutoML**-Pipelines, die viele Algorithmen ausprobieren. Wie oben erwähnt, ist in Materials aber „viel Daten“ relativ – man wird selten in den fünfstelligen Bereich kommen. Falls doch: dann würde man vermutlich schon Big-Data-Techniken nutzen, etwa verteiltes Training oder aufwändige Deep Learning Modelle wie **Transformers** (auch denkbar: es gibt erste Ansätze, chemische Formeln mit Transformer-Modellen als Sequenz zu behandeln, wobei die Sequenz die Bausteine repräsentiert).

Metriken und Validierung: Für die Bewertung der Modelle muss man die richtigen Metriken definieren. Bei Regressionszielen wie Härte oder Aushärtetiefe eignen sich RMSE und R^2 am besten, um einerseits die durchschnittliche Abweichung zu sehen und andererseits die erklärte Varianz. Oft werden in Materialpapers auch **Relative Errors** angegeben (z.B. „Durchschnittsfehler = 5%“), was bei heteroskedastischen Daten sinnvoll ist (denn ein absoluter Fehler von 2 könnte bei einer Härte von 10 gravierend sein (20%), bei einer Härte von 100 aber vernachlässigbar (2%)). Daher könnte man MAE auch ins Verhältnis zum Wert setzen. Für Klassifikationen (druckbar/nicht druckbar) ist bei stark unausgeglichenen Klassen die **F1-Score** oder der **Matthews-Korrelationskoeffizient** besser als plain Accuracy, um sicherzugehen, dass die Minderheitsklasse nicht untergeht. In unserem Fall – druckbar vs. nicht – ist es vermutlich essentiell, die nicht-druckbaren richtig zu identifizieren (denn eine

nichtdruckbare Rezeptur im Druckversuch ist sehr teuer). Also wäre die **Precision** auf „nicht druckbar“ (Verlässlichkeit, dass ein als „nicht druckbar“ vorhergesagter wirklich nicht druckt) und die **Recall** (wie viele der tatsächlich nicht druckbaren werden erkannt) zu beachten.

Für die Optimierung wiederum braucht es eigene Metriken: Ein Kriterium ist die **Anzahl der Experimente bis zum Ziel**. Wenn z.B. ein Bayesian Optimization nach 5 Iterationen eine acceptable Rezeptur findet, ist das besser als nach 15. Man kann auch **Regret** messen – die Differenz zwischen aktuellem Bestwert und dem unbekannten globalen Optimum – allerdings kennt man letzteres nicht, man kann nur angeben, wie schnell der beste gefundene Wert steigt. In Multi-Objective-Problemen nutzt man oft den **Hypervolumen-Indikator**, der misst, welchen Volumenanteil im Zielraum die Pareto-Front abdeckt; dieser sollte mit Iterationen steigen. Für Simulation-optimierung (in silico) kann man vergleichen, wie die vom Algorithmus vorgeschlagenen Rezepturen abschneiden gegenüber z.B. randomisierter Suche. Und für die generelle **Praxisbewertung** muss letztlich ein Domänenexperte prüfen: Sind die vorgeschlagenen Rezepturen chemisch sinnvoll, praktikabel (Kosten, Verfügbarkeit der Stoffe) und reproduzierbar? Diese weichen Kriterien fließen zwar nicht direkt in ML-Metriken ein, sollten aber qualitativ mitbetrachtet werden. Hier kann eine Ontologie auch helfen, indem sie z.B. *Praktikabilitäts-Wissen* enthält (etwa: Stoff X ist toxisch oder teuer, meide wenn möglich).

Sinnvolle hybride Kombinationen

Basierend auf dem bisher Diskutierten lassen sich einige Kombinationen von Methoden identifizieren, die **wissenschaftlich fundiert und praxistauglich** erscheinen, um unser Problem anzugehen. Ein paar exemplarische Hybridansätze:

- **Ontologie-gestütztes Feature Engineering + Baumensemble:** Dies wäre ein relativ einfach umzusetzender Start. Man nutzt die Ontologie, um zusätzliche Merkmale pro Rezeptur zu erzeugen (Klassenanzahl, Summen, bekannte Einflussfaktoren) und trainiert dann z.B. einen Gradient Boosting Decision Tree (wie XGBoost). Entscheidungsbäume können mit gemischten numerischen und kategorischen Features umgehen und profitieren von sinnvollen Features. Das Modell wäre schnell trainiert, leicht interpretierbar (Feature Importances, Regelwege) und könnte als Surrogat in einem BO-Loop dienen (es gibt Ansätze mit Random Forest als Surrogat, sog. Treed GPs etc.). Diese Kombi setzt wenig Daten voraus und nutzt Expertenwissen voll aus – ideal in frühen Projektphasen.

- **Gaussian Process + physikalische Priors:** Hier würde man z.B. für eine bestimmte Zielgröße wie Aushärtetiefe ein GP-Modell aufsetzen, dem man einen Mean-Prior in Form eines bekannten physikalischen Modells gibt. Denkbar: das Beer-Lambert-Modell (Belichtungsenergie nimmt exponentiell mit Eindringtiefe ab, abhängig vom Absorbergehalt). Dies als Baseline und GP lernt die Abweichungen davon (vielleicht verursacht durch Füllstoffstreuung etc.). So fließt Theorie ins ML ein. GPs können Priors aufnehmen und diese durch Daten anpassen [13] [45]. Kombiniert mit Bayesian Optimization hätte man so eine „Physics-informed BO“. Erste Arbeiten in anderen Domänen zeigen, dass physik-informierte GPs schneller konvergieren und realistischere Optima liefern, weil sie den Suchraum einschränken.

- **Graph Neural Network + Knowledge Graph Embedding:** Diese Kombination adressiert das komplexe Eingabeformat. Man könnte parallel zwei Graphen darstellen: (1) einen **Material-Wissensgraph** (aus der Ontologie: Knoten = Substanzen, Kanten = „istA“, „verträglichMit“, etc.), (2) pro Rezeptur einen **Mischungsgraph**, der Verbindungen zwischen den Komponenten im konkreten Rezept hat (man könnte z.B. einen vollständigen Graphen annehmen, wo jeder Stoff mit jedem verbunden ist, oder bipartit: Stoffe <-> Formulierungs-Knoten). Ein GNN könnte beide

Graphen schichtweise verarbeiten, möglicherweise mit verschiedenen Message-Passing-Regeln, und am Ende eine gemeinsame Einbettung erzeugen, aus der dann die Eigenschaften vorhergesagt werden. Dies ist aufwändig und erfordert Forschung, aber es vereint Daten und Wissen hochgradig. *Praxis:* Noch haben wenige Gruppen so etwas umgesetzt, aber Bausteine sind vorhanden (z.B. MatKG als Materials Knowledge Graph [69] [70], PolymerGNN für Mischungen [71] [72]).

- **Symbolic Regression + Ontologie-Constraints:** Man könnte versuchen, mit genetischem Programming Formeln zu finden, aber das Suchverfahren mit logischen Regeln einschränken. Z.B. nur Formelbestandteile zulassen, die in der Domäne sinnvoll sind (etwa lineare Terme für additive Effekte, quadratische Terme um Synergien zweier Komponenten abzubilden, exponentielle Terme für Absorptionsprozesse). Die Ontologie könnte hier herangezogen werden, um z.B. automatisch Formelteile zu generieren: *„(Monomer_Acrylat_Gehalt * Photoinitiator_Gehalt)“* etc. So würde symbolische Regression effizienter suchen im Raum der plausible Gleichungen. Vorteil: Das Resultat wäre eine lesbare Formel, die man chemisch interpretieren kann. Nachteil: symbolische Regression ist rechnerisch teuer und bei vielen Variablen schwierig.

- **Bayesian Optimization + aktives Lernen + Simulation:** Falls Simulationsmodelle existieren (z.B. rheologische Simulation der Suspension oder optische Simulation der Belichtung), könnte man diese nutzen, um einen Teil der Experimente virtuell durchzuführen. Multi-Fidelity-Optimierung ist ein Gebiet, das niedriggenaue aber billige Datenquellen (Simulation) mit hochgenauen teuren Quellen (Realexperiment) kombiniert [73] [38]. Ein hybrider Plan wäre, initial synthetische Datenpunkte aus der Simulation zu ziehen, das ML-Modell vorzutrainieren, und dann mittels Bayesian Optimization nur wenige reale Experimente gezielt dort durchzuführen, wo Simulationen unsicher sind. Dadurch spart man Laborzeit. Diese Idee erfordert natürlich, dass solche Simulationen existieren und ausreichend genau sind, was in unserem Fall je nach Aspekt variieren mag.

Viele der genannten Kombinationen sind *wissenschaftlich plausibel*. Die praktische Umsetzbarkeit hängt von verfügbarem Aufwand und Daten ab. In einem industriellen Umfeld könnte man iterativ vorgehen: **Start einfach, dann iterativ erweitern.** Beginnen mit einem gut gepflegten Datensatz und einem einfachen Modell (z.B. Random Forest), um einen Grundstock an Ergebnissen zu haben. Dann Ontologiemerkmale hinzufügen, schauen ob’s verbessert. Dann vielleicht GPR einführen für Unsicherheitsschätzung. Dann das Experiment-Loop mit BO initialisieren, um gezielt neue Daten zu sammeln (wobei parallel das Modell immer wieder neu trainiert wird). Mit wachsender Datenbasis kann man in komplexere Deep-Learning-Modelle oder GNNs investieren, die dann mehr Nuancen lernen (wie spezifische chemische Strukturenffekte). Dieses sukzessive Vorgehen stellt sicher, dass der Modellaufbau nachvollziehbar bleibt und man aus jedem Schritt Domänen-Einsicht zieht (z.B. welche Features wichtig sind, welche Kombinationen das Modell vorschlägt – das sind oft Learnings über die Materialchemie selbst).

Methodenauswahl: Entscheidungsbaum

Abschließend lässt sich die Auswahl der richtigen Methodik als Entscheidungsprozess darstellen. Im Folgenden ein konzeptioneller **Entscheidungsbaum**, der je nach Projektsituation zu einer Empfehlung führt:

1. **Datenumfang beurteilen:**
2. **Sehr wenige Daten (< ~50 Samples):** Setze auf **wissensgetriebene Modelle**. → Nutzen von einfachen Modellen mit starker Regularisierung (z.B. lineare Modelle mit Feature-Engineering,

SVM mit geeignetem Kernel, GPR). Integriere Domainwissen maximal: manuell definierte Features, Constraints. Gegebenenfalls zusätzlich Simulationen/Experten-Schätzungen verwenden, um Datenpunkte zu ergänzen. **Deep Learning ist hier nicht ratsam.**

3. **Moderate Daten (ca. 50–500):** → Wähle **klassische ML-Algorithmen** (Random Forest, Gradient Boosting, SVM, kleine neuronale Netze) und kombiniere sie bei Bedarf mit Transfer Learning oder Ensemble-Techniken. Prüfe, ob Ontologiewissen als Features einbindbar ist (das verbessert oft die Generalisierung). Beginne eventuell schon mit **Bayesian Optimization** oder Active Learning, um gezielt neue Punkte zu sammeln – hier glänzt GPR oder Random Forest als Surrogat, da Unsicherheitsschätzungen verfügbar sind.

4. **Große Datenmenge (> 500, vor allem > 1000):** → Erwäge **tiefere neuronale Netze** oder spezialisierte Architekturen (GNNs, Transformer), da genügend Daten eine komplexere Modellierung rechtfertigen. Standard-Algorithmen wie XGBoost werden zwar immer noch gute Leistung bringen, aber ein gut entworfenes Deep Model könnte nun Mehrwert bieten (z.B. automatische Merkmalsextraktion aus Molekülstrukturen). Achte dennoch auf Validierung – auch 1000 Punkte können bei einem sehr großen Rezepturraum noch wenig sein. Ontologieintegration kann hier auch über fortgeschrittene Methoden laufen (Knowledge Graph Embeddings ins NN). Große Daten erlauben auch splitting: vielleicht separate Modelle für verschiedene Subräume (Clustering der Formulierungen und je Cluster ein Modell).

5. **Zieltyp klären (Vorhersage vs. Optimierung):**

6. **Nur Vorhersage benötigt:** → Fokus auf **überwachte Lernmodelle** mit bestmöglicher Präzision. Hier sind Ensemble-Methoden oft Top, also ggf. ein **Stacking oder Voting Ensemble** aus mehreren Algorithmen verwenden, um Generalisierung zu verbessern [40]. Wenn Interpretierbarkeit gewünscht: lineare Modelle oder wenige Tree-Depth Bäume einsetzen, oder komplexes Modell mit Explainable-AI-Methoden analysieren. Mehrere Outputs? → Multi-Task-Netz oder separate Modelle pro Output (je nach Korrelation der Outputs).

7. **Optimierung gewünscht:** → Falls ein Vorhersagemodell existiert, nutze es als Surrogat und führe **In-silico-Suche** durch (GA, Exhaustive Search, etc.). Falls Experimente iterativ geplant werden sollen, setze eine **Bayesian Optimization**-Schleife auf (anfangs mit einem einfachen Surrogatmodell, z.B. GPR). Multi-Objective? → Wähle einen Algorithmus, der Pareto-Lösungen liefert (z.B. Multiobjektiv-BO oder GA mit Multi-Objective-Ranking). Wichtig: Bei Optimierung immer Unsicherheit mit einbeziehen, um Fehlschläge zu vermeiden – hier ist ein probabilistisches Modell von Vorteil [74]. Plane genügend Experimente für die Exploration ein (ggf. zunächst breite Suche, später Feinjustierung – kann auch algorithmisch eingestellt werden [47] [48]).

8. **Verfügbares Expertenwissen berücksichtigen:**

9. **Ontologie/Regeln verfügbar:** → Diese unbedingt nutzen. Wenn Daten klein sind, kann Wissen sogar wichtiger als fancy Algorithmus sein. Also Ontologie-basierte Features kreieren, Unzulässige Kombinationen in Suche ausschließen, evtl. **knowledge-driven priors** in Bayesian Model oder initial population für GA definieren (z.B. starte GA mit paar heuristisch guten Rezepturen vom Experten). Algorithmen, die Wissen leichter aufnehmen: GPR (via Prior), Bayesian Networks (via Strukturvorgabe), ILP (via Hintergrundwissen), dedizierte neuro-symbolische Methoden.

10. **Wenig explizites Wissen vorhanden:** → Dann datengetrieben vorgehen. Algorithmen mit guter Mustererkennung (Random Forest, Neural Nets) in den Vordergrund stellen. Trotzdem: implizites Wissen aus Literatur einfließen lassen z.B. durch informierte Initialisierungen der Optimierung

(Start mit Literatur-Rezepturen, um Modell schnell auf sinnvolles zu bringen). Und sobald etwas gelernt ist, versuchen, *ex post* aus dem Modell Regeln abzuleiten (Feature Importance, Partial Dependence Plots, etc.), was wiederum neues Wissen generiert.

11. **Modell komplexität vs. Daten prüfen:**

12. Eher zu einfaches Modell als zu komplex, wenn unsicher. Man kann mit einem einfachen Modell Baselines erstellen – wenn diese schon gute Performance liefern (z.B. $R^2 > 0.8$), braucht man kein kompliziertes Deep Learning erzwingen. Umgekehrt, wenn einfache Modelle systematisch scheitern (z.B. $R^2 \sim 0.3$ trotz aller Features), kann es sein, dass komplexere Zusammenhänge vorliegen, die evtl. ein tieferes Netz oder spezielle Kernels nötig machen. Hier könnte man eine **Fehleranalyse** machen: schauen, ob die Residuen irgendeinem Muster folgen, das Hinweise gibt (z.B. immer unterschätzt bei Formulierungen mit Additiv X -> vielleicht fehlt eine Interaktion im Modell; ein Netz könnte die lernen).

13. **Iterativ verbessern:**

14. Nach initialer Modellauswahl und Training: Validierungsergebnisse auswerten. Falls Performance ungenügend: Schrittweise Optimieren – Datenqualität verbessern (Ausreißer entfernen, mehr Daten sammeln), Feature-Satz erweitern (Ontologie intensiver nutzen, neue Messdaten hinzufügen falls möglich), oder Algorithmus wechseln (wenn Random Forest zu unflexibel scheint, mal GPR probieren, etc.). Die Entscheidungsfindung ist also kein einmaliger Baum, sondern ein Regelkreis.

In Form eines **vereinfachten Entscheidungsdiagramms**:

- **Start:** Wenig Daten? → Wähle small-data-freundliche Algorithmen (GP, RF, SVM) + Knowledge-Features. **Sonst:** Bei viel Daten → fortgeschrittene Algorithmen (Deep Learning, GNN).
- **Ziel:** Optimierung nötig? → Setze BO/Active Learning ein (ggf. auf Basis des gewählten Vorhersagemodells). **Sonst (nur Vorhersage):** Konzentriere dich auf Modellgüte und ggf. Interpretierbarkeit.
- **Wissen vorhanden?** → Integriere Ontologie in Features/Prior/Constraints. **Kein Wissen:** Nutze rein datengetriebene Patterns, aber sei vorsichtig mit Extrapolation.

Dieser Entscheidungsprozess soll sicherstellen, dass die eingesetzten Methoden stets **angemessen zum Problem** sind – man möchte weder „mit Kanonen auf Spatzen schießen“ (z.B. ein überdimensioniertes Deep Learning bei Mini-Datensatz), noch Potenzial verschenken (z.B. Domainwissen ignorieren, obwohl es große Lücken füllen könnte).

Fazit und Ausblick

In dieser Übersicht haben wir die Landschaft der potenziell geeigneten ML-Modelle, Optimierungsverfahren und hybriden Ansätze für die Vorhersage und Optimierung von 3D-Druck-Resinformulierungen mit Keramikpartikeln dargestellt. **Klassische Algorithmen** (wie Random Forests oder SVM) bieten robuste Leistung bei kleinen Daten und leichter Interpretierbarkeit, **probabilistische Modelle** (v.a. Gaussian Processes) ermöglichen Unsicherheitsabschätzung und sind ideal für aktive Lernansätze in der Laboroptimierung [13] [45]. **Deep Learning**, insbesondere spezialisierte Formen wie **Graph Neural Networks**, eröffnet Wege, komplexe Zusammenhänge und chemische Strukturen zu erfassen – dies entfaltet seine Stärke aber erst bei ausreichend Daten oder muss durch Transfer Learning und Wissenseinspeisung unterstützt werden [15] [16]. **Symbolische Methoden und Wissensinjektion** sorgen dafür, dass das reichhaltige verfügbare Expertenwissen und die

Materialontologie nicht ungenutzt bleiben, sondern die Modelle leiten und absichern. Gerade **hybride Lösungen** – etwa die Kombination von Wissensgraphen mit maschinellem Lernen – sind ein aktueller Trend, um das Beste aus beiden Welten zu vereinen [28] [60]. Für praktische Anwendungen in der Materialentwicklung ist zudem die Schleife aus **Vorhersage und Experiment** entscheidend: mittels **Bayesian Optimization** und aktivem Lernen können ML-Modelle und Laborversuche Hand in Hand neue optimale Formulierungen erschließen, wie jüngste Studien eindrucksvoll gezeigt haben [1] [2] [20] [21].

Aktuelle Veröffentlichungen der Jahre 2020–2025 untermauern die Machbarkeit und den Nutzen dieser Ansätze. Die Arbeiten reichen von speziellen Anwendungen (z.B. ML-gestützte Optimierung von Photopolymerkompositen für mechanolumineszente Sensoren [49] [50] oder aktives Lernen für UV-härtende Acrylate [20] [56]) bis hin zu Überblicksartikeln, die generelle Methoden für kleine Daten in der Materialforschung diskutieren [3] [4]. Diese zeigen auch auf, dass das *Zusammendenken* verschiedener KI-Techniken der Schlüssel ist, um komplexe Probleme wie Materialformulierung zu bewältigen. Kein einzelner Algorithmus ist allen Situationen gewachsen [75] [76], aber durch eine sinnvolle Kombination – ausgewählt entlang klarer Kriterien wie Datenvolumen, Zielsetzung und Wissensverfügbarkeit – lässt sich ein leistungsfähiges System gestalten.

Für die Praxis in Forschung und Entwicklung bedeutet dies, dass man zunächst die Grundlagen (saubere Daten und ontologisches Wissensmodell) schaffen sollte, um dann iterativ mit ML zu arbeiten: erst vorhersagen, validieren, dann optimieren und wieder neuen Erkenntnissen lernen. Dieses **wissenschaftlich fundierte, iterative Vorgehen** kann die Entwicklung optimaler 3D-Druck-Formulierungen erheblich beschleunigen und zu Materialien mit maßgeschneiderten Eigenschaften führen, die über konventionelle Trial-and-Error-Methoden nur schwer gefunden würden.

Quellen: Die im Text referenzierten Studien und Übersichtsarbeiten (2020–2025) sind in den eckigen Klammern angegeben, z.B. [1] verweist auf einen Ausschnitt der jeweiligen Publikation. Diese Belege untermauern die gemachten Aussagen mit aktuellen Forschungsergebnissen und bieten Ansatzpunkte für vertiefende Lektüre.

---

[1] [2] [51] [52] [2106.15697] Accelerated Discovery of 3D Printing Materials Using Data-Driven Multi-Objective Optimization
https://arxiv.org/abs/2106.15697

[3] [4] [5] [6] [7] [12] [13] [14] [17] [18] [19] [30] [31] [32] [33] [34] [35] [36] [37] [38] [39] [41] [42] [43] [45] [46] [47] [48] [58] [59] [64] [74] [75] [76] Small data machine learning in materials science | npj Computational Materials
https://www.nature.com/articles/s41524-023-01000-z?error=cookies_not_supported&code=5225ee84-f4af-46dc-bf8f-9932b9ae54ad

[8] [9] [10] [11] [49] [50] [53] [54] [55] Machine Learning in 3D and 4D Printing of Polymer Composites: A Review
https://www.mdpi.com/2073-4360/16/22/3125

[15] [16] [22] [23] [44] [67] [68] [71] [72] Polymer graph neural networks for multitask property learning | npj Computational Materials
https://www.nature.com/articles/s41524-023-01034-3?error=cookies_not_supported&code=27fbb290-5c31-4818-a2e2-1190fe2de56b

[20] [21] [56] [57] Machine-Guided Discovery of Acrylate Photopolymer Compositions - PubMed
https://pubmed.ncbi.nlm.nih.gov/38534124/

24 25 26 27 28 29 60 69 70 Material Property Prediction with Element Attribute Knowledge Graphs and Multimodal Representation Learning
https://arxiv.org/pdf/2411.08414

40 ACS Applied Polymer Materials Vol. 6 No. 8
https://pubs.acs.org/toc/aapmcd/6/8

61 Ontology-based feature engineering in machine learning workflows ...
https://www.nature.com/articles/s41598-022-23101-3

62 73 Data quantity governance for machine learning in materials science
https://pmc.ncbi.nlm.nih.gov/articles/PMC10265966/

63 Domain knowledge-assisted materials data anomaly detection ...
https://www.sciencedirect.com/science/article/pii/S2352847825000565

65 Recent progress on machine learning with limited materials data
https://www.sciencedirect.com/science/article/pii/S2352847824001552

66 Recent progress on machine learning with limited materials data
https://www.sciopen.com/article/10.1016/j.jmat.2024.07.002