Digital Discovery

ROYAL SOCIETY OF CHEMISTRY

PAPER

View Article Online
View Journal | View Issue

Cite this: *Digital Discovery*, 2025, **4**, 2765

Decoding non-linearity and complexity: deep tabular learning approaches for materials science

Vahid Attari * and Raymundo Arroyave

Materials datasets, particularly those capturing high-temperature properties pose significant challenges for learning tasks due to their skewed distributions, wide feature ranges, and multimodal behaviors. While tree-based models like XGBoost are inherently non-linear and often perform well on many tabular problems, their reliance on piecewise constant splits can limit effectiveness when modeling smooth, long-tailed, or higher-order relationships prevalent in advanced materials data. To address these challenges, we investigate the effectiveness of encoder–decoder model for data transformation using regularized Fully Dense Networks (FDN-R), Disjunctive Normal Form Networks (DNF-Net), 1D Convolutional Neural Networks (CNNs), and Variational Autoencoders, along with TabNet, a hybrid attention-based model, to address these challenges. Our results indicate that while XGBoost remains competitive on simpler tasks, encoder–decoder models, particularly those based on regularized FDN-R and DNF-Net, demonstrate better generalization on highly skewed targets like creep resistance, across small, medium, and large datasets. TabNet's attention mechanism offers moderate gains but underperforms on extreme values. These findings emphasize the importance of aligning model architecture with feature complexity and demonstrate the promise of hybrid encoder–decoder models for robust and generalizable materials prediction from composition data.

Received 22nd April 2025
Accepted 7th July 2025

DOI: 10.1039/d5dd00166h

rsc.li/digitaldiscovery

1. Introduction

Predicting materials behavior is inherently challenging due to the non-linear and interdependent relationships between alloy chemistry, processing conditions, and properties. Tabular materials data often include dense numerical and sparse categorical features with weaker correlations and multi-modal characteristics, making pattern recognition more difficult. Deep learning has found success in tasks such as sequence-to-sequence modeling and data reconstruction, and shows promise in capturing complex dependencies for modeling materials properties. However, applying deep learning models to materials data presents unique challenges. In materials science, physics-based relationships require explicit incorporation of domain knowledge, and the "black-box" nature of these models limits interpretability. Their success also hinges on the availability of large, high-quality datasets, which are often costly and scarce. Furthermore, the significant computational demands of deep learning raise questions about its efficiency compared to simpler methods like random forests or physics-informed neural networks. Addressing these challenges is essential to fully harness the value of materials data for advancing predictive modeling, improving design efficiency, and complementing traditional physics-based approaches (Fig. 1).[1–7]

Deep learning has shown remarkable promise across various domains, including materials science, where it has been successfully applied to predict mechanical properties of alloys,[6,8,9] discover new thermoelectric materials,[7] and identify phase transitions in complex multicomponent systems.[10] However, while these successes highlight its potential, applying deep learning to tabular data, which is prevalent in materials science, presents unique challenges. Materials science data often spans multiple orders of magnitude, reflecting the diversity of material properties and phenomena. For example, mechanical properties like yield strength range from tens of MPa for polymers to thousands of MPa for metals and ceramics, while electrical conductivity varies from as low as 10^{-16} S m^{-1} in insulators to 10^{7} S m^{-1} for conductors like copper. Similarly, thermal conductivity can range from less than 0.1 W m^{-1} K^{-1} in insulators to over 1000 W m^{-1} K^{-1} in materials like diamond, and diffusion coefficients vary from 10^{-25} m^2 s^{-1} in solids at low temperatures to 10^{-8} m^2 s^{-1} in liquids. Even creep behavior spans orders of magnitude under high-temperature conditions, influenced by factors such as stress, temperature, and microstructure.

Data scarcity, challenges in data preparation, predictive accuracy, and interpretability are all vital considerations in materials science applications to ensure that models provide actionable insights for experimental validation. To tackle data

Department of Materials Science & Engineering, Texas A&M University, College Station, 77840, TX, USA. E-mail: attari.v@tamu.edu

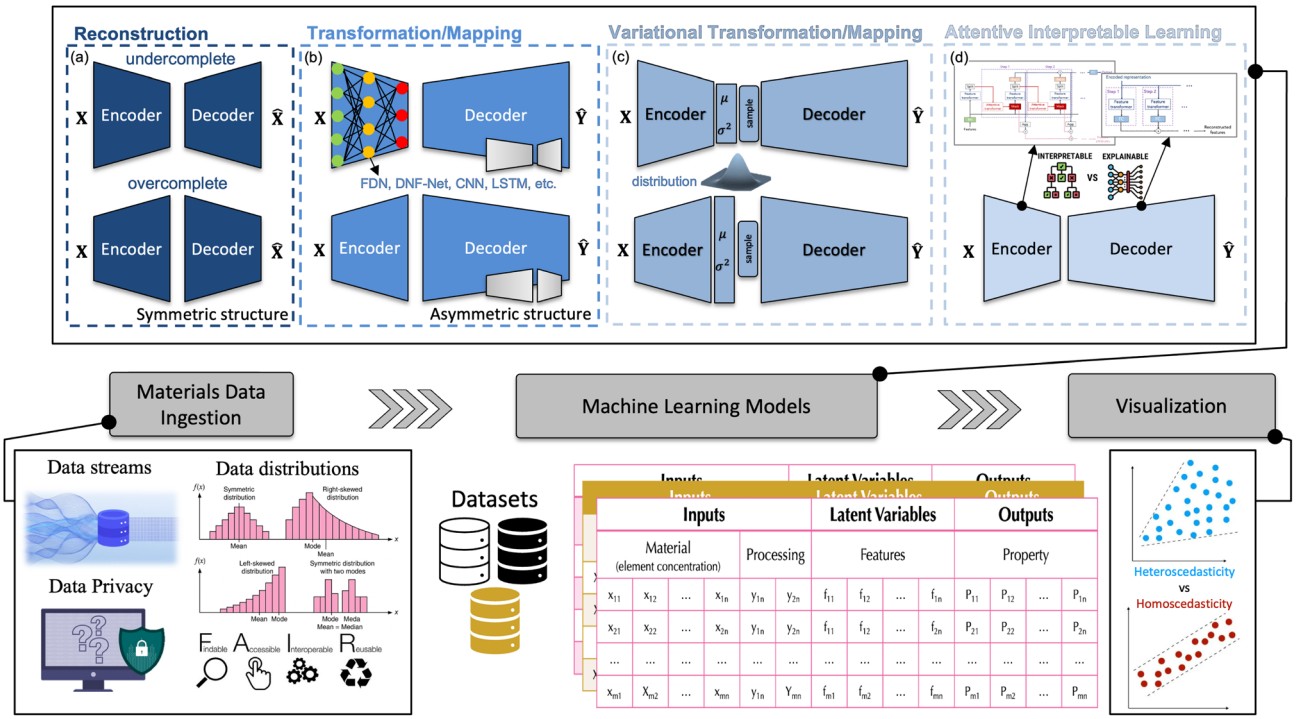


Fig. 1 Framework for materials data processing and insight generation. The diagram illustrates the complete workflow from data ingestion through machine learning to visualization. The materials data ingestion panel (bottom left) shows data streams, privacy considerations, and statistical distributions of ingested materials data. The datasets and tables (center) explicitly represent how the processed data, including elemental compositions, processing parameters, and derived features are structured and passed into the machine learning models. The top panels depict different encoder–decoder architectures: (a) autoencoder models for reconstruction tasks, (b) asymmetric encoder–decoder models (FDN-R, DNF-Net, CNN) for feature transformation and mapping, (c) variational encoder–decoder models for handling uncertainty, and (d) encoder–decoder models with attention mechanisms for interpretability. The workflow culminates in the visualization panel (far right), highlighting property prediction insights such as heteroscedasticity analysis. Together, this framework integrates data reconstruction, transformation, uncertainty quantification, and interpretability to support robust property prediction and materials informatics analysis.

scarcity and confidentiality concerns, generative approaches like variational autoencoders (*e.g.* Tab-VAE[11]) or Tabular Generative Adversarial Networks[12] can be employed to create synthetic datasets, enhancing model robustness. Tackling data preparation for largely skewed features can be achieved using robust transformations, such as quantile transformation or log-scaling, to ensure features are more suitable for model training.[13,14] When paired with interpretable architectures[15] and efficient inference strategies,[16] these methods can advance the use of tabular materials data in scientific discovery and design. Recent innovations, such as transformers and hybrid models combining tree-based methods with neural networks, show promise in overcoming these challenges. Transformers leverage self-attention mechanisms to capture complex feature interactions, offering an advantage for the heterogeneous and multi-scale nature of materials data. However, inconsistencies in benchmarking practices and unequal levels of optimization hinder direct comparisons with traditional methods like gradient-boosted decision trees (GBDT).[17–21] To fully realize their potential, deep learning models in materials science must balance predictive accuracy with alignment to domain-specific physical principles and interpretability. Achieving this will require improved benchmarking frameworks, access to diverse and high-quality datasets, and the integration of domain knowledge into model development. These advancements could close the performance gap with traditional methods and enable deep learning to address the unique demands of materials science and unlock its potential for solving complex problems.

Several emerging neural architectures, inspired by concepts from the encoder–decoder and other hybrid methods, have shown promise in addressing the limitations of traditional models, particularly in handling tabular data workflows. For example, TabNet enhances interpretability and performance by dynamically selecting relevant features through attention mechanisms.[22] Neural Oblivious Decision Ensembles (NODE) combine decision rules with neural networks to efficiently model feature interactions.[17] FT-Transformers and TabTransformer employ self-attention to capture complex relationships between numerical and categorical features.[23,24] Hybrid approaches like DeepGBM integrate gradient-boosting machines with neural networks to combine feature transformation and prediction.[25] These advancements, along with models like Wide and Deep Networks[26] and DNF-Net (Disjunctive Normal Form Networks),[27] improve scalability, interpretability, and predictive performance, positioning them as robust alternatives to traditional methods.

In this work, we evaluate the potential of several deep learning architectures to outperform traditional GBDTs in accuracy and efficiency when applied to materials science tabular datasets. Our findings demonstrate that neural architectures tailored for tabular data can effectively handle the complexity and wide range of property scales often encountered in materials science. Beyond static datasets, these models exhibit promise in processing streams of data generated from high-throughput experiments or autonomous systems, making them particularly relevant for real-time analysis. Additionally, their ability to capture intricate dependencies and handle complex, multimodal distributions positions them as critical tools for addressing the inherent variability in materials data. These advancements also align with the principles of FAIR (Findable, Accessible, Interoperable, and Reusable) data, ensuring that the insights derived from these models can be leveraged across diverse research and industrial contexts. In Section 2, we detail the architectures and approaches used in this study, while Section 4 compares the performance of deep learning models and GBDTs, highlighting their respective strengths in handling complex materials datasets.

2. Methods for tabular data

2.1. Baseline encoder–decoder pipeline

The encoder–decoder model is widely recognized for its ability to reconstruct data and capture complex relationships between features by learning a latent representation. For regression tasks, it can be adapted to learn complex input space and predict continuous output values. The encoder maps the input data $\mathbf{X}$ into a latent space $\mathbf{Z}$, which can either have a lower-dimensional representation (undercomplete model) or a higher-dimensional representation (overcomplete model) for rich representative learning. The decoder then maps $\mathbf{Z}$ to the predicted output $\hat{\mathbf{Y}}$. These transformations are defined as:

$$\mathbf{Z} = f_\theta(\mathbf{X}),\ \hat{\mathbf{Y}} = g_\phi(\mathbf{Z}), \tag{1}$$

where the model is trained by minimizing a loss function L, which measures the difference between the predicted output $\hat{\mathbf{Y}}$ and the target data $\mathbf{Y}$. The objective is to find the parameters θ and ϕ that minimize the following:

$$\min_{\theta,\phi} \mathrm{L}\left(\mathbf{Y}, \hat{\mathbf{Y}}\right) = \mathrm{L}\left(\mathbf{Y}, g_\phi(f_\theta(\mathbf{X}))\right) \tag{2}$$

Training is typically performed using gradient-based methods. After training, the model can be used to transform new data $\mathbf{X}_{\mathrm{new}}$ into predictions $\hat{\mathbf{Y}}_{\mathrm{new}}$:

$$\hat{\mathbf{Y}}_{\mathrm{new}} = g_\phi(f_\theta(\mathbf{X}_{\mathrm{new}})) \tag{3}$$

This baseline encoder–decoder approach is not inherently interpretable, primarily due to the lack of transparency in its latent representations. However, with attention mechanisms, feature analysis tools, and visualizations, it can become explainable, providing some insights into its decision-making process. Next, we will discuss three neural network architectures: regularized fully connected dense networks, Disjunctive Normal Form Networks (DNF-Nets), and Convolutional Neural Networks (CNN). These architectures, when combined with encoder–decoder frameworks, can enhance prediction accuracy by leveraging latent space representations and reducing noise. Interpretability is further improved through feature selection in DNF-Nets and filter visualization in 1D-CNNs. Interpretability ensures trust and accountability by enabling the understanding of model predictions, feature importance, and decision-making processes, which is particularly crucial for debugging models, achieving regulatory compliance, and gaining user acceptance.[28,29]

2.1.1. Regularized dense neural network block. A fully dense neural network, also known as a fully connected or feedforward network, is a foundational deep learning architecture that remains widely used for solving diverse machine learning problems across various dataset types.[30] In this architecture, each neuron in one layer connects to every neuron in the next, allowing the model to capture complex, non-linear relationships between features. For a given input vector $\mathbf{x}$, the output of the i-th layer is computed as:

$$\mathbf{y}_i = f(\mathbf{W}_i\mathbf{x} + \mathbf{b}_i) \tag{4}$$

where $\mathbf{W}_i$ is the weight matrix, $\mathbf{b}_i$ the bias term, and f a non-linear activation function such as ReLU, sigmoid, or tanh. Multiple layers refine the data representation, improving model predictions. The parameters θ include the weight matrices, biases, and, if applicable, trainable activation parameters like the slope α in LeakyReLU. Regularization terms, such as the L2 coefficient λ, help control overfitting by penalizing large weights. These parameters collectively determine how the dense block processes and learns from data. Fully connected layers are not transparent, offering little insight into their decision-making process.[31]

2.1.2. Disjunctive normal form (DNF) block. A DNF network block is a neural network architecture designed to model logical structures, similar to decision trees, making it interpretable and efficient for capturing complex relationships in tabular data. The DNF block operates by combining multiple logical conjunctions (AND conditions), which are then aggregated using disjunctions (OR conditions). Each conjunction represents feature interactions modeled through linear transformations and activation functions, mimicking the behavior of a logical AND operation. The outputs of these conjunctions are then aggregated to form a disjunction, effectively modeling a logical OR operation. This hierarchical structure enables the DNF block to identify and represent complex feature relationships in a manner that is both interpretable and computationally efficient. Key parameters of a DNF block include the number of clauses (representing distinct logical interactions), the number of literals per clause (defining how many features contribute to each interaction), the choice of activation function (*e.g.*, ReLU or Sigmoid), and regularization techniques such as dropout to prevent overfitting. This structure makes DNF particularly suitable for applications requiring interpretability and robust modeling of feature interactions.

2.1.3. 1D-convolutional neural network block. A 1D-Convolutional Neural Network (1D-CNN) block is specifically designed to extract local patterns from sequential or ordered data, such as time series or audio signals.[32] The architecture employs convolutional filters that slide across the input data, capturing localized features through a combination of linear transformations and non-linear activation functions. These filters enable the model to detect patterns such as trends or periodicities, which are essential for understanding structured data. Typically, the convolutional layers are followed by pooling layers that down-sample the data, reducing computational complexity while preserving critical information. This hierarchical structure allows 1D-CNNs to progressively capture more abstract patterns as the data flows through deeper layers. Parameters of the model include the weights of the convolutional filters, biases, and the choice of activation functions, such as ReLU, which introduces non-linearity to the model. 1D-CNNs can be combined with fully connected layers to refine feature representations, making them versatile for a variety of tasks. Recent advancements have demonstrated the efficacy of these neural network blocks in competitions and real-world applications by leveraging shortcut connections and innovative architectures to improve feature extraction and accuracy, particularly in domains where spatial locality is less pronounced, such as tabular data.

2.2. TabNet: attentive interpretable tabular learning

TabNet employs a sequential attention mechanism to dynamically select the most relevant features at each decision step, enabling it to efficiently capture complex relationships in tabular datasets.[22] Unlike traditional fully connected neural networks, TabNet performs feature selection dynamically, mirroring the interpretability of decision tree models while retaining the flexibility and power of neural networks. The TabNet encoder is composed of a feature transformer, an attentive transformer and feature masking. A split block divides the processed representation to be used by the attentive transformer of the subsequent step as well as for the overall output. TabNet decoder is composed of a feature transformer block at each step. Attention mechanisms have emerged as a pivotal architectural element in deep neural networks (DNNs), significantly enhancing their interpretability.[31] This mechanism ensures that the outputs from one step guide the feature selection process for subsequent steps. By iteratively refining feature selection and transformation across multiple steps, this model's parameter set encompasses weight matrices, bias terms, and parameters for the attention mechanism, which collectively determine the feature selection and transformation process. To prevent overfitting, regularization techniques such as L2 regularization and sparsity-inducing penalties are commonly applied.

2.3. Generative tabular learning using VAE

The deterministic design of classical encoder–decoder models limits their ability to generate diverse outputs, constraining their effectiveness in capturing and navigating complex data distributions. A VAE shares an encoder–decoder structure, but extends it by incorporating probabilistic modeling in the latent space, making it ideal for cases where uncertainty in predictions is important and generative tasks. The encoder transforms the input $\mathbf{X}$ into a distribution over the latent space, rather than a single point. Specifically, the encoder outputs the mean μ_i and variance ${\sigma_i}^2$ for the latent variable $\mathbf{z}$:

$$\mathbf{z}_i = \mu_i + \sigma_i \odot \varepsilon, \quad \varepsilon \sim \mathcal{N}(0,1) \tag{5}$$

where $\mathbf{z}_i$ represents a latent variable sampled from the distribution, μ_i and σ_i are learned parameters that describe the mean and variance, and ε is a random noise sampled from a standard normal distribution, enabling stochasticity in the latent space. The decoder then maps the data by mapping the latent variable $\mathbf{z}_i$ back to the output space:

$$\hat{\mathbf{y}}_i = f_{\text{decoder}}(\mathbf{W}_i\mathbf{z}_i + \mathbf{b}_i) \tag{6}$$

where f_{decoder} is a non-linear transformation (*e.g.*, a fully connected layer with ReLU), $\mathbf{W}_i$ represents the weight matrix for the decoder, and $\mathbf{b}_i$ is the bias term.

The objective of the variational encoder–decoder is to maximize the evidence lower bound (ELBO), which consists of two terms: the reconstruction loss (*e.g.*, mean squared error between the input $\mathbf{x}$ and the reconstruction $\hat{\mathbf{y}}$) and the Kullback–Leibler (KL) divergence between the learned latent distribution and a standard normal distribution. The full loss function can be written as:

$$\mathrm{L} = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{y}|\mathbf{z})] - D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})) \tag{7}$$

The parameter set θ in a variational encoder–decoder block includes the weights and biases of the encoder and decoder, the mean and variance parameters μ_i and σ_i, and any regularization terms used to control overfitting.

2.4. Extreme gradient boosting

Extreme Gradient Boosting (XGBoost) is a powerful ML algorithm based on decision trees, optimized for speed and performance. It belongs to the family of boosting algorithms, where multiple weak learners (typically decision trees) are combined sequentially to form a strong predictive model.[33] XGBoost works by fitting a new tree to correct the errors of the previous trees, iteratively improving the model's predictions. Given an input vector $\mathbf{x}$, the output of the model at step t is computed as:

$$\hat{y}_{\mathrm{t}} = \sum_{i=1}^{t} f_i(\mathbf{x}) \tag{8}$$

where $f_i(\mathbf{x})$ represents the prediction from the i-th tree, and $\hat{\mathrm{y}}_{\mathrm{t}}$ is the cumulative prediction at step t. The trees are added one by one to minimize the residual errors of the previous trees. XGBoost optimizes a regularized objective function that consists of a loss term and a regularization term to prevent overfitting:

$$\mathrm{L} = \sum_{i=1}^{n} \ell(y_i, \hat{y}_i) + \sum_{k=1}^{T} \Omega(f_k) \tag{9}$$

where $\ell(y_i, \hat{y}_i)$ is the loss function, which measures the difference between the true label y_i and the predicted label $\hat{\mathrm{y}}_i$, and

$\Omega(f_k)$ is a regularization term that controls the complexity of each tree, penalizing large trees to avoid overfitting. The parameter set θ for XGBoost includes the weights of the decision trees, the regularization coefficients (L_1, L_2), and the learning rate, which controls how quickly the model adapts to errors during training where L_1 (lasso) encourages sparsity by penalizing the absolute sum of weights, and L_2 (ridge) discourages large weights by penalizing the squared sum.

3. Implementation and optimization of models

3.1. Bayesian hyperparameter optimization

The predictive performance and generalization ability of ML models, particularly deep learning architectures, are highly sensitive to their hyperparameters.[34] Examples include the number of layers, learning rate, dropout rate, and regularization parameters. To systematically explore these, we employed the tree-structured Parzen estimator (TPE), a versatile Bayesian optimization algorithm, implemented *via* the Optuna framework.[35–38] The objective was to minimize the loss function $f(x)$:

$$x_{\text{opt}} \in \arg\min_{x \in X} f(x) \tag{10}$$

where x_{opt} denotes the set of hyperparameters yielding the best performance. Compared to exhaustive grid or random search, TPE efficiently concentrates sampling in promising regions of the hyperparameter space, significantly reducing computational cost.

We performed hundreds of optimization trials per dataset, tuning between 5 and 10 hyperparameters depending on the architecture. These parameters and their range are listed in Table S4. Each trial was evaluated over a fixed training duration of 50 epochs (or number of boosting rounds for XGBoost), rather than using early stopping, to ensure fair comparison of convergence behavior. To test the significance of performance differences across models, we applied Friedman's test at a 95% confidence level.

Prior to optimization, we quantified first- and higher-order statistics of each feature, such as mean, standard deviation, skewness (asymmetry), and kurtosis (tailedness). These moments informed our expectations for model learning challenges, such as handling heavy-tailed or skewed distributions.

To evaluate generalization, we applied a standard random split strategy, typically reserving 10–15% of the data as a hold-out test set, with the remaining data used for training and internal validation during hyperparameter optimization. In cases where the dataset was small, the validation and test sets occasionally overlapped, serving as a pragmatic compromise to assess performance. K-fold cross-validation was subsequently applied using the best hyperparameters to rotate the test split, offering a more comprehensive assessment and reducing variance in performance estimates.

3.2. Evaluation metrics

Model performance was evaluated using a range of regression metrics designed to capture different aspects of prediction error, tailored to the challenges of materials datasets. For targets spanning multiple orders of magnitude, we employed scale-normalized metrics such as the Mean Squared Logarithmic Error (MSLE) and the Symmetric Mean Absolute Percentage Error (SMAPE) to balance contributions across scales. Standard measures like Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) were also reported to maintain comparability with common benchmarking practices.

We note that while SMAPE is valuable for normalizing errors relative to the combined magnitude of predictions and true values, it introduces slight asymmetry that can favor over-predictions. To complement accuracy-focused metrics, we further assessed optimization efficiency and stability through average convergence rates, calculated as the mean relative improvement in loss across successive trials, highlighting how quickly each model approached its best performance. Table 1 summarizes the formal definitions of these metrics.

3.3. Hyperparameter importance

To further understand the interplay between data characteristics and model design, we analyzed the importance of each hyperparameter in driving performance improvements. Hyperparameter importance was computed using functional ANOVA (fANOVA) as implemented in Optuna, which estimates the expected change in the objective function when varying one hyperparameter while holding others near their optimal values. This helps identify which hyperparameters most influence model loss, guiding future tuning efforts and clarifying why certain architectures may exhibit greater sensitivity to optimization (as evidenced by steeper convergence rates or more variable trial outcomes).

By comparing hyperparameter importance profiles across models and datasets, we could directly connect these insights back to our earlier statistical characterization (skewness, kurtosis, and combined complexity measure $|\text{skew}| + |\text{kurt} - 3|$), illustrating how data distribution properties impacted the relative tuning needs of different architectures.

4. Results and discussion

4.1. Datasets: ATLAS-RHEA, BIRDSHOT, and MPEA

The ATLAS-RHEA dataset[39] comprises comprehensive computational and experimental data for 10 626 refractory HEA compositions, emphasizing creep behavior and thermophysical properties. It includes 71 features, such as elemental compositions (*e.g.*, Nb, Cr, V, W, Zr), thermal/mechanical properties (*e.g.*, coefficients of thermal expansion, thermal conductivity), and phase stability metrics across various temperatures. The dataset also covers creep metrics over time intervals (25–2000 minutes) and temperatures (1300 K–2000 K), along with Kou Criteria for phase stability. While minor data gaps exist in some properties (*e.g.*, PROP LT, PROP ST, Kou Criteria, ~11% incomplete entries), the dataset provides a valuable resource for exploring structure–property relationships and optimizing high-performance refractory alloys.

Table 1 Evaluation metrics highlighting various aspects of prediction accuracy in materials informatics

Metric	Formula
Mean squared logarithmic error (MSLE)	$\frac{1}{n}\sum_{i=1}^{n}(\log(y_i+1)-\log(\hat{y}_i+1))^2$
Root mean squared logarithmic error (RMSLE)	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}\log(y_i+1)-\log(y_i+1)^2}$
Logarithmic coefficient of determination ($\log R^2$)	$1-\frac{\sum(\log y_i-\log \hat{y}_i)^2}{\sum(\log y_i-\overline{\log y})^2}$
Geometric mean absolute error (GMAE)	$\exp\left(\frac{1}{n}\sum_{i=1}^{n}\lvert\log y_i-\log \hat{y}_i\rvert\right)$
Symmetric mean absolute percentage error (SMAPE)	$\frac{100\%}{n}\sum_{i=1}^{n}\frac{\lvert y_i-\hat{y}_i\rvert}{\lvert y_i\rvert+\lvert\hat{y}_i\rvert/2}$
Mean absolute scaled error (MASE)	$\frac{1}{n}\sum_{i=1}^{n}\frac{\lvert y_i-\hat{y}_i\rvert}{\frac{1}{n-1}\sum_{j=2}^{n}\lvert y_j-y_{j-1}\rvert}$
Root mean squared percentage error (RMSPE)	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i-\hat{y}_i}{y_i}\right)^2}$

Table 2 summarizes key first-order statistics, highlighting skewness and kurtosis as important factors influencing model performance. For instance, the Cobalt creep resistance at 1300 °C in the ATLAS-RHEA dataset exhibits extreme skewness (39) and kurtosis (1500), reflecting a highly non-normal distribution with substantial outliers. These properties underscore the necessity for targeted pre-processing techniques, such as log or quantile transformations. Additionally, robust models capable of managing extreme outliers may be required. In contrast, features with lower skewness and kurtosis, such as yield strength 1000 °C or pugh ratio, display distributions closer to normality, making them suitable for direct use in training without extensive preprocessing. These features are less likely to cause instability in the trained models and typically contribute to more reliable and consistent predictions. Features with moderate skewness and kurtosis, such as Scheil LT and Kou criteria, suggest some degree of asymmetry and outliers, which could still impact model performance if not accounted for during pre-processing.

The BIRDSHOT dataset[40] currently contains detailed information on 147 non-equimolar Cantor high-entropy alloys (HEAs), focusing on their composition, processing parameters, and mechanical properties. Each alloy is characterized by its elemental fractions (Al, Co, Cr, Cu, Fe, Mn, Ni, V) and evaluated for key properties such as YS, ultimate tensile strength (UTS), tension elongation, and hardness. The dataset also includes computationally derived parameters such as stacking fault energy, valence electron concentration, and the Pugh ratio, providing insights into the alloys' mechanical behavior and stability. For example, the yield strength of the alloys ranges from 310 MPa to 537 MPa, while tension elongation varies from 18.3% to 25.7%. First-order statistics for this dataset is summarized in Table 3.

Despite its comprehensive nature, the dataset includes minor missing data across key properties. Measured properties like YS and UTS have 141 complete entries (~96%), while computed parameters such as stacking fault energy, valence electron concentration, and hardness are available for 131

Table 2 ATLAS-RHEA dataset: first-order statistical summary including skewness and kurtosis of some features

Feature	Mean	Std. dev.	Min	Max	Median	25%	75%	Skewness	Kurtosis	Complexity
Nb	0.22	0.19	0.00	0.95	0.20	0.05	0.35	0.909	3.289	1.20
Cr	0.21	0.18	0.00	0.95	0.15	0.05	0.30	0.955	3.458	1.41
V	0.22	0.19	0.00	0.95	0.20	0.05	0.35	0.940	3.400	1.34
W	0.14	0.11	0.00	0.90	0.10	0.05	0.20	0.699	3.143	0.84
Zr	0.21	0.18	0.00	0.90	0.15	0.05	0.30	0.915	3.300	1.21
YS 1000 °C	1170.58	637.06	0.00	3399.29	1111.17	673.36	1599.91	0.478	2.742	0.74
EQ 1273 K THCD (W mK^{-1})	14.11	12.95	0.02	62.92	9.27	3.43	22.70	0.974	3.009	0.98
EQ 1273 K density (g cm^{-3})	9.59	1.77	6.05	16.94	9.33	8.29	10.65	0.773	3.563	1.34
1300 min creep CB (1 s^{-1})	0.00	0.10	0.00	4.01	0.00	0.00	0.00	**39.012**	**1524.244**	**1560.26**
PROP 1500 °C CTE (1 K^{-1})	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.124	3.353	1.48
YS 1500 °C PRIOR	807.87	497.41	0.00	2689.29	738.53	415.22	1127.56	0.656	2.995	0.66
Pugh ratio PRIOR	2.60	0.51	1.45	4.52	2.56	2.23	2.92	0.390	2.914	0.48
SCHEIL LT	1620.39	163.87	1507.01	2353.37	1541.34	1529.62	1599.97	1.954	5.858	4.81
Kou criteria	353.63	402.90	0.13	4656.12	240.34	62.72	463.10	2.184	9.764	8.95
Creep merit	77 876.76	431 904.11	−825.33	10 135 045.82	507.71	40.83	6176.66	11.793	199.473	208.27

Table 3 Birdshot dataset: first-order statistical summary including skewness and kurtosis for experimental data, contained 147 rows on March 2025

Feature	Mean	Std. dev.	Min	Max	Median	25%	75%	Skewness	Kurtosis	Complexity
Al	1.68	2.34	0.00	15.00	0.00	0.00	4.00	1.564	8.058	6.62
Co	18.57	14.24	0.00	75.00	16.00	8.00	25.00	1.014	4.235	2.25
Cr	7.84	6.32	0.00	25.00	8.00	4.00	10.00	0.875	3.225	1.10
Cu	1.63	3.88	0.00	24.00	0.00	0.00	2.00	3.531	16.961	17.49
Fe	15.07	11.89	0.00	75.00	15.00	5.00	20.00	1.686	7.613	6.30
Mn	5.33	8.10	0.00	40.00	0.00	0.00	8.00	1.602	5.256	3.86
Ni	39.28	14.23	0.00	75.00	40.00	31.00	50.00	−0.521	3.051	0.57
V	10.59	8.17	0.00	30.00	10.00	4.00	15.50	0.545	2.252	1.29
Yield strength (MPa)	388.41	125.27	176.82	790.00	367.00	282.19	460.50	0.630	2.802	0.83
UTS true (MPa)	952.40	239.12	333.00	1581.00	958.00	802.50	1096.62	0.012	2.867	0.15
Elongation (%)	33.65	12.12	0.00	55.70	34.00	26.10	42.25	−0.627	3.307	0.93
Hardness (GPa) SRJT	2.52	0.67	1.63	6.04	2.39	2.09	2.77	2.516	12.689	12.21
Modulus (GPa) SRJT	196.84	22.44	141.59	263.51	193.38	180.73	212.70	0.216	2.798	0.42
Avg HDYN/HQS	1.15	0.04	1.06	1.25	1.15	1.13	1.17	−0.107	3.008	0.11
Depth of penetration (mm) FE	2.86	0.27	2.22	3.46	2.87	2.66	3.05	−0.140	2.429	0.71

samples (∼89%). While the dataset remains highly usable, handling these missing values through imputation or focusing on well-reported features will be critical for accurate analysis and predictive modeling. The dataset contains 22 features, capturing both experimental and computational results, and is suitable for developing predictive models, investigating structure–property relationships, and advancing the discovery of novel high-performance alloys.

The Multi-Principal Element Alloy (MPEA) dataset[41] contains information on 1545 alloys across 23 columns, focusing on chemical composition, microstructure, processing methods, and mechanical properties. Key attributes include alloy formula, microstructure classification (*e.g.*, FCC, BCC, or mixed phases), grain size, processing methods (*e.g.*, casting), and mechanical properties such as hardness (HV), YS, UTS, elongation, and Young's modulus. The dataset also includes test conditions (*e.g.*, temperature, test type), computationally derived parameters (*e.g.*, calculated density and Young's modulus), and bibliographic details such as DOI references and publication metadata. The first-order statistics about this dataset is summarized in Table S3.

The dataset exhibits significant variability in data completeness, with properties like grain size (∼15%), hardness (∼34%), and elongation (∼40%) moderately reported, while others, such as experimental density (∼7%) and Young's modulus (∼9%), are highly sparse. Elemental contamination data (*e.g.*, oxygen, nitrogen, and carbon content) are particularly limited, with only 57, 45, and 4 entries, respectively. These data gaps may introduce biases and limit usability for certain analyses. Well-reported features like yield strength (∼69%) and processing methods can be prioritized for modeling, while missing values in moderately sparse features may be addressed using imputation. Highly sparse features, such as carbon content, should be treated cautiously or excluded from predictive tasks to ensure result integrity.

4.2. Hyperparameter optimization: ATLAS-RHEA data

We selected the Kou criterion for single-feature hyperparameter optimization due to its relatively high statistical complexity, characterized by notable skewness (2.2) and kurtosis (6.8). These properties make it a challenging yet informative objective, which is advantageous for evaluating model's ability to explore a broader and less symmetric parameter space effectively. Fig. 2a shows the variance in model performance during hyperparameter optimization, comparing multiple encoder–decoder regressor architectures with VAE and XGBoost conducted using BO. The *y*-axis represents the objective value on a log scale, while the *x*-axis shows the number of trials, sorted by their objective performance. The goal is to minimize the loss, with lower curves indicating better optimization results. The individual convergence plots (Fig. 2a) reveal model-specific responses to BO, offering insights beyond the sorted view. The optimization speed and behavior vary significantly across the models. Both the FDN-R and DNF models exhibit rapid initial improvements, with sharp drops in objective values within the first 20–30 trials, quickly converging to stable solutions. In contrast, models like VAE and XGBoost show slower, more gradual improvements across trials, requiring more extensive exploration to identify optimal configurations. This variation in convergence speed has practical implications: while FDN-R and DNF can quickly find strong solutions with fewer trials, models like VAE and XGBoost benefit from longer optimization runs to thoroughly explore their parameter spaces. Understanding these differences is crucial for efficiently allocating computational resources during hyperparameter tuning, particularly when scaling to larger datasets.

Table 4 highlights the tailored hyperparameter configurations across models. To reduce the hyperparameter search space and streamline optimization, the number of layers in both the encoder and decoder is initially fixed. This strategy is effective when a baseline architecture demonstrates satisfactory performance. As model development advances, relaxing this constraint allows the architecture to adapt to data-specific complexity and potentially enhance generalization. This flexibility becomes particularly important when optimizing models for datasets with varying feature dimensions and complexity, such as BIRDSHOT and MPEA. As Table 4 shows the FDN-R and

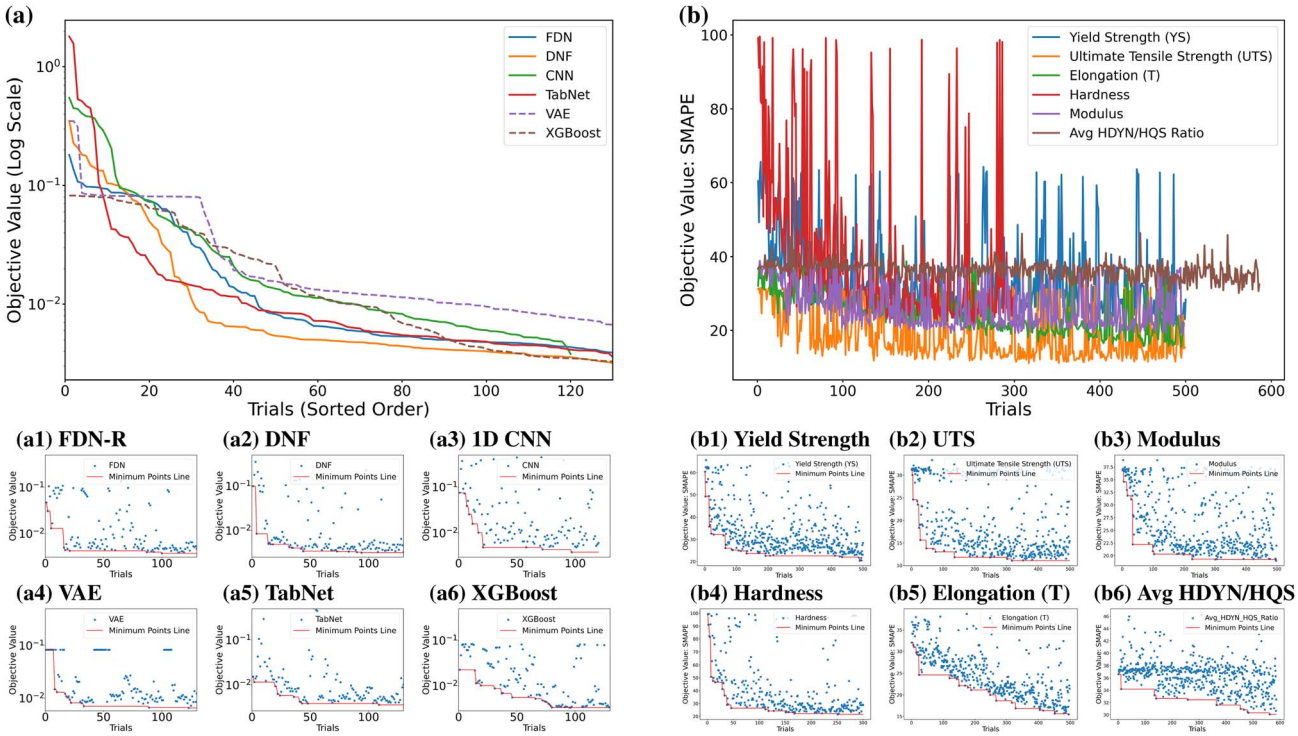


Fig. 2 Bayesian hyperparameter optimization history across models (left) based on ATLAS-RHEA data and target properties (right) based on BIRDSHOT data. Embedded labels (a1–a6 and b1–b6) correspond to individual model or target plots. (a) ATLAS-RHEA dataset: optimization history by model. (a) Shows sorted trials across all models. (a1–a6) Correspond to FDN-R, DNF, 1D CNN, VAE, TabNet, and XGBoost, respectively. (b) Birdshot HEA dataset: optimization history by target feature. (b) Shows combined trial history, followed by (b1–b6): yield strength, UTS, elastic modulus, hardness, elongation, and HDYN/HQS ratio.

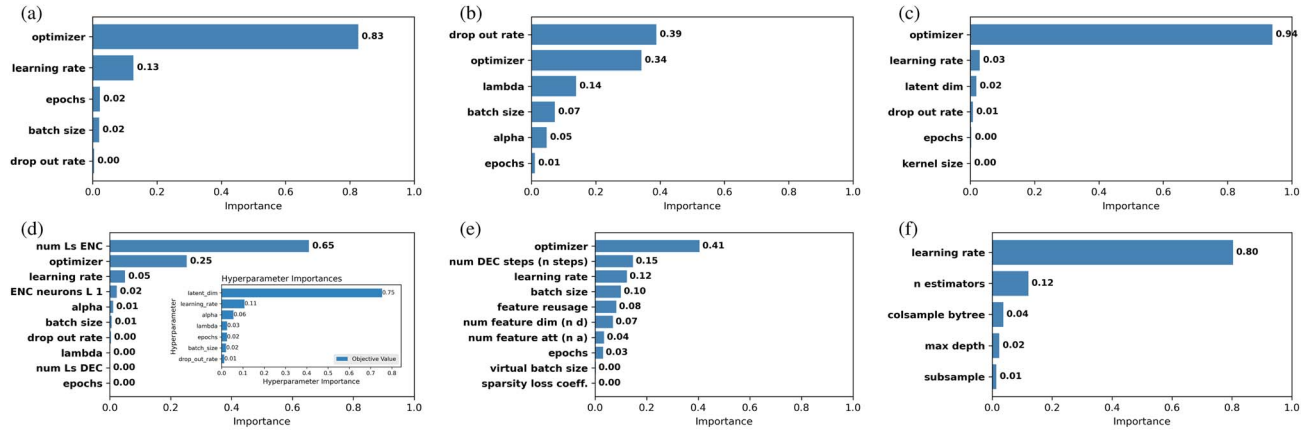
Table 4 Best Hyperparameters for different models using ATLAS-RHEA dataset

Study name	Layers	Latent dim	Drop-out rate	Learning rate	Optimizer	Batch size	Epochs	Additional hyperparameters
Regularized FDN-R	Fixed	192	0.1	1.26×10^{-4}	adam	96	50	$\lambda = 3.69 \times 10^{-6}$, $\alpha = 0.0164$
DNF		64	0.1	5.39×10^{-4}	adam	32	50	n_conj = 10, conj_units = 112
CNN		64	0.1	3.77×10^{-4}	adam	32	50	Kernel size = 4
TabNet		—	—	4.24×10^{-3}	RMSprop	72	50	$n_d = 24$, $n_a = 32$, $\lambda = 7.98 \times 10^{-4}$
VAE		16	0.2	3.58×10^{-4}	adam	128	50	$\lambda = 2.87 \times 10^{-5}$, $\alpha = 0.045$
XGBoost		—	—	9.58×10^{-2}	—	—	—	n_estimators = 50, max depth = 10

DNF employ smaller learning rates ($\sim 10^{-4}$) compared to the higher rates used by TabNet and XGBoost ($\sim 10^{-2}$). Batch sizes also vary, with DNF using smaller batches (32), while VAE benefits from larger batches (128). Regularization terms (λ) are applied in FDN-R, DNF, and VAE to prevent overfitting. Adam is the preferred optimizer for most models, except for TabNet, which benefited using RMSprop. Additionally, architecture-specific parameters like conjunction units (112) are critical for DNF performance.

Fig. 2b illustrates the target-wise hyperparameter optimization history for the BIRDSHOT HEA dataset using the FDN-R model. Rather than optimizing across multiple models, the focus here is on individual target properties, such as yield strength, ultimate tensile strength, elastic modulus, hardness, elongation, and the HDYN/HQS ratio, each treated as a distinct regression task. The objective value, SMAPE, exhibits different convergence behaviors depending on the target, highlighting the varying difficulty and noise levels associated with predicting each property. This target-based optimization approach is more appropriate in this context, as it allows fine-tuning of the FDN-R model architecture and learning parameters to the specific statistical characteristics of each property, thereby maximizing predictive performance due to small nature of this dataset and ensuring model generalization across diverse features.

Fig. 3 shows the hyperparameter importance for all investigated models. The hyperparameter importance quantifies the average influence of each hyperparameter on the optimization objective (loss), estimating how much the loss increases when



(g)

Model	Best Loss Value ↓	Optimizer	Best Trial Duration	Test R^2	Avg. Convergence Rate (%)
Encoder-Decoder (DNF-Net)	**0.00316**	adam	29.39	0.962	-232.7
XGBoost	0.00327	–	0.63	0.960	-75.2
Encoder-Decoder (VAE)	0.00333	adam	89.35	0.958	-32.5
TabNet	0.00358	RMSprop	152.17	–	-95.0
Encoder-Decoder (FDN-R)	0.00374	adam	330.67	–	-74.6
Encoder-Decoder (1D-CNN)	0.00379	adam	324.58	0.954	-212.1

Fig. 3 Ranking of hyperparameter importance for different models, where importance reflects the average increase in the optimization objective (loss) when each hyperparameter is perturbed from its best value. Panels show results for: (a) DNF, (b) FDN-R, (c) CNN, (d) VAE, (e) TabNet, and (f) XGBoost. In panel (d), the inset plot is a separate trial conducted with a fixed optimizer to show the sensitivity of the VAE model to other hyperparameters. (g) The enclosed table summarizes the best loss values, selected optimizers, trial durations, test set R^2, and average convergence rates for each model trained on the "Kou Criteria" feature. Convergence rates appear as negative because they represent reductions in loss across trials; larger magnitudes indicate faster decreases but may also suggest increased sensitivity to hyperparameter tuning.

that hyperparameter is varied across its sampled range while others are held near optimal. Each subplot shows the relative importance of various hyperparameters in determining model performance, providing insights into which parameters have the greatest impact on the objective value during optimization. For example, the DNF and FDN-R models (Fig. 3a and b) highlight distinct hyperparameters as the most influential, whereas the CNN and VAE models (Fig. 3c and d) demonstrate different sensitivity patterns. The variation in hyperparameter importance across models underscores the unique tuning needs of each architecture.

Among the hyperparameters tuned, the optimizer choice often emerged as particularly influential, likely due to its direct role in controlling how model weights are updated during training. Optimizers like Adam and RMSprop can significantly influence convergence speed, training stability, and the quality of the final solution by adapting learning rates and managing gradients effectively. This adaptability often makes the optimizer more impactful than other hyperparameters like dropout rate or batch size, which primarily regulate model complexity or regularize training without fundamentally altering the learning process.

For FDN-R architecture, the type of optimizer (Adam, SGE, Adadelta) has significant impact on the model performance, contributing 100% to the objective value's reduction. If we fix the optimizer to adam, the most important hyperparameters are the latent dimension contributing 75%, learning rate contributing 11%, and negative slope of activation function contributing 6%, and the rest of the parameters show minor importance. The FDN-R model shows consistent improvement across multiple runs with the best hyper-parameters, with progressively lower validation losses. Friedman's test reveals a significant difference in performance between the runs, with a test statistic of 191.176 and a p-value of 1.11×10^{-18}, indicating that the observed performance variations are statistically meaningful. This suggests that factors such as random initialization or data splits may influence the model's performance, warranting further investigation to enhance consistency. Alternatively, for VAE, the results of Friedman's test, with a test statistic of 28.70 and a p-value of 0.052, suggest that there is no statistically significant difference in the performance of the model across multiple training runs at a 95% confidence level. Although the p-value is close to the threshold of 0.05, it is slightly higher, indicating that we cannot reject the null hypothesis of no significant difference. This implies that the variations observed in the validation loss across the runs are likely due to random chance rather than meaningful differences in the model's performance.

The enclosed table in Fig. 3 offers key insights into model performance, highlighting variations in optimization efficiency, test R^2 values, and average convergence rates. We define the convergence rate as the average relative improvement in the objective value across successive trials, calculated by $\frac{L_{i-1} - L_i}{L_{i-1}} \times 100$, where L_i denotes the loss at trial i. A larger magnitude of this metric (appearing as more negative in our plots, reflecting sharper decreases) indicates faster reductions in loss, typically when the hyperparameter search discovers

narrow optimal regions. While such behavior accelerates convergence, it may also imply increased sensitivity to specific hyperparameter settings, which could impact both the robustness of optimization and the reproducibility of scientific results. In many high-throughput or industrial settings, consistent and stable model behavior, without extensive hyperparameter retuning, is essential to maintain both operational efficiency and scientific precision. Models that depend heavily on finely tuned parameters, reflected by large convergence jumps, may therefore be less suitable for rapid deployment pipelines.

Among the models evaluated, XGBoost emerges as particularly well-suited for high-throughput or industrial applications. It achieves a competitive loss value (0.00327) and solid test set R^2, while demonstrating the shortest trial duration (0.63 seconds) and a moderate convergence rate (−75.2%) that suggests reliable optimization without reliance on extremely narrow hyperparameter configurations. In contrast, although the DNF model attains the lowest loss (0.00316) and highest test R^2 (0.962), its steep convergence trajectory (−232.7%) and greater sensitivity to hyperparameter selection could complicate use in workflows that demand reproducibility and minimal tuning. TabNet and VAE, while offering valuable features such as interpretability or probabilistic flexibility, are characterized by longer trial durations and less stable convergence behavior, which may limit their practicality in time-sensitive applications. These insights help guide model selection by emphasizing the trade-offs between predictive performance, optimization stability, computational demands, and the need for scientific accuracy across repeated experiments.

This relationship is evident across the models evaluated. The DNF model stands out with the lowest loss value (0.00316) and the highest test R^2 (0.962), demonstrating strong predictive capabilities. However, its large magnitude of convergence rate (−232.7%) combined with variability across optimization trials suggests greater sensitivity to hyperparameter choices, which may complicate its application in high-efficiency contexts. XGBoost, with a slightly higher loss (0.00327), exhibits exceptional computational efficiency, achieving the shortest trial duration (0.63 seconds), making it well-suited for large-scale datasets where time is a critical factor. Despite its speed, XGBoost's convergence rate (−75.2%) still indicates moderate optimization variability over extended trials. TabNet, while competitive in loss value (0.003586), shows a substantially longer trial duration (152.17 seconds) and a convergence rate of −95.0%, suggesting that its attention mechanism, although valuable for interpretability, adds computational overhead. The VAE, while achieving reasonable test R^2 scores, faces challenges with prolonged trial durations and less consistent convergence behavior, indicating inefficiencies in both computational cost and optimization robustness. These findings suggest that DNF and XGBoost are particularly promising for applications requiring a balance of predictive accuracy and efficiency, whereas models such as VAE and TabNet, with their longer runtimes and sensitivity to optimization, may be better suited for exploratory or interpretability-focused studies. These insights help guide model selection by highlighting the trade-offs between predictive performance, stability, and computational demands.

4.3. Generalization of models to complex features

This section evaluates how different models generalize to output features of varying statistical complexity in the ATLAS-RHEA dataset. Parity plots in Fig. 4 compare the predictive performance of the XGBoost, regularized FDN-R, and 1D CNN models across four features: YS 1000 °C, EQ 1273 K density, Kou Criteria, and 1300 Min Creep CB. Extended comparisons, including TabNet, DNF-Net, and VAE, are provided in Fig. S4. Model complexity increases from top to bottom, while feature complexity increases from left to right. Model complexity was evaluated based on structural metrics such as the number of trainable parameters for neural networks (regularized FDN-R, DNF-Net, VAE, 1D CNN) and the number of trees, maximum depth, and total nodes/splits for the XGBoost model. Feature complexity was calculated as the sum of the absolute skewness and the absolute excess kurtosis ($|\text{skewness}| + |\text{kurtosis} - 3|$) for each output feature. The displayed metrics, including MSLE, RMSLE, $\log R^2$, GMAE, SMAPE, MASE, and RMSPE, capture various aspects of generalization.

The main observation in Fig. 4 is that as both model complexity and feature complexity increase, predictive performance varies significantly across models. While XGBoost (a) performs well for relatively simple features such as yield strength and density, its performance deteriorates for the most complex feature, 1300 Min Creep CB, indicating limited generalization to highly skewed or outlier-heavy data. In contrast, the encoder–decoder model based on regularized FDN-R architecture maintains strong performance across all features and demonstrates superior generalization on the most complex output, achieving lower error metrics and higher alignment with the parity line. The 1D CNN, despite being the most complex model, fails to consistently improve performance and struggles with both simple and complex features, suggesting that higher model complexity does not always translate to better generalization, especially for skewed or outlier-prone data. This highlights the importance of appropriately balancing model complexity with feature distribution characteristics.

For features like YS 1000 °C and density, both exhibiting low feature complexity, models such as XGBoost, regularized FDN-R, and DNF-net show strong predictive performance with low error metrics and high $\log R^2$ values. However, as feature complexity increases with Kou Criteria and 1300 Min Creep CB, model behavior diverges. Among all models, the regularized FDN-R shows the highest robustness and generalization to this severe skewness, achieving an R^2 of 0.856 and SMAPE of 56.8% on the Creep feature, outperforming all other methods including XGBoost and VAE. This indicates that the regularized FDN-R's architectural simplicity combined with its ability to regularize through sparsity allows it to handle extreme data distributions effectively, even outperforming models that typically exhibit higher expressiveness.

While models like CNN begin to struggle with Kou Criteria and Creep due to the presence of extreme values and long tails in the distribution, the regularized FDN-R and DNF-net encoder–decoder models retain relatively stable performance

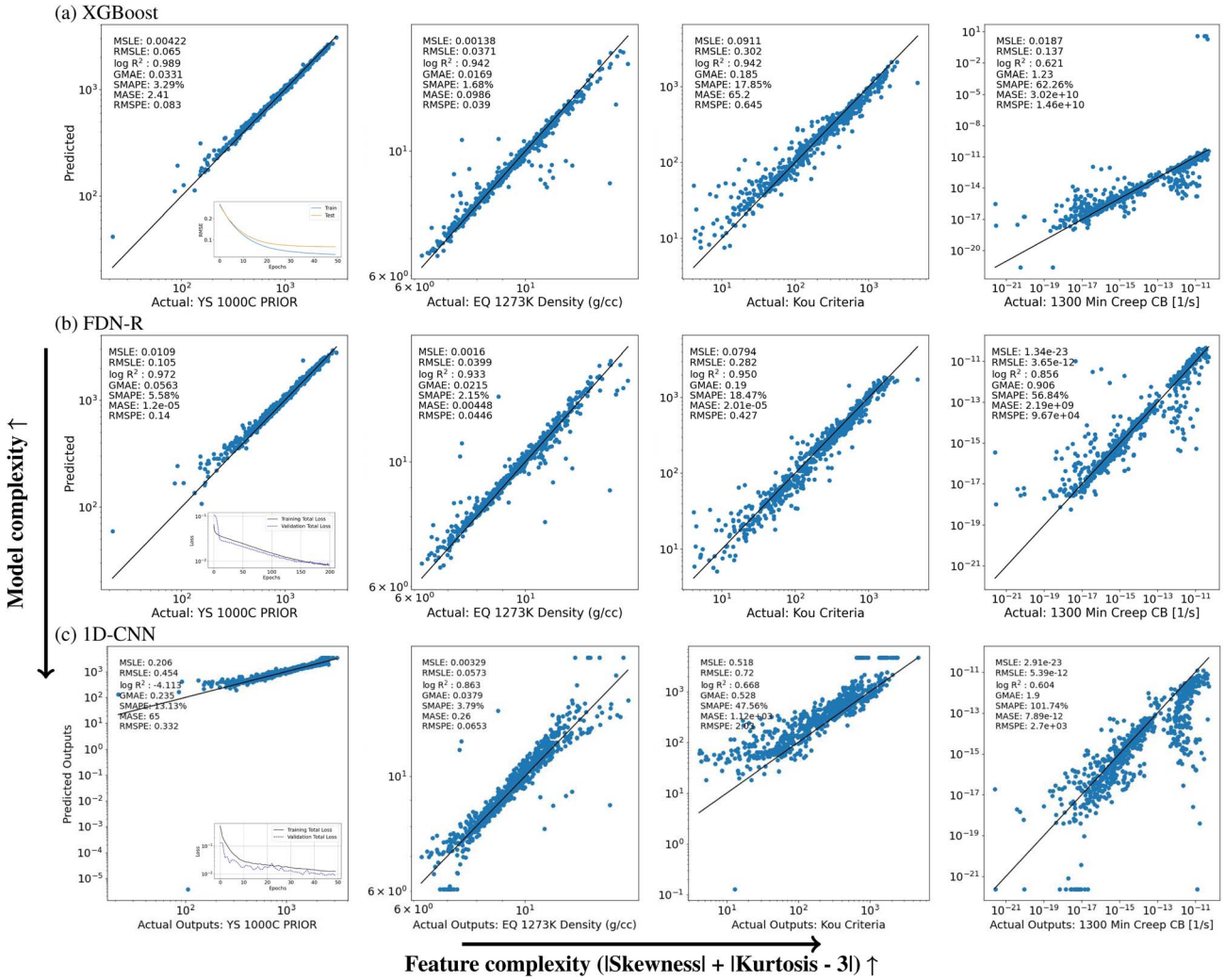


Fig. 4 Parity plots comparing the predictive performance of models (a) XGBoost, (b) regularized fully dense network ((1437381 model parameters)), and (c) 1D convolutional neural network (1D CNN) across four output features from the ATLAS-RHEA dataset. Model complexity increases from top to bottom (XGBoost → regularized FDN-R → 1D CNN), while feature complexity increases from left to right, corresponding to yield strength (0.73), density (1.33), Kou criteria (9.94), and creep (1560.0). For the complete set of parity plots and clear training history, refer to Fig. S4 in the SI document. The smaller insets in each row display the training and validation history corresponding to each model and are available in the SI document.

across all feature complexities. XGBoost and VAE models also show notable resilience across the board, particularly on skewed features, though it slightly underperforms the regularized FDN-R for Creep. In contrast, 1D CNN model exhibit sharp performance degradation on highly skewed features, highlighting the limitations of more complex or generative architectures in handling rare or extreme data cases without additional regularization or specialized training strategies.

Skewed distributions, especially with long tails, pose significant challenges for regression models. They can violate assumptions such as residual normality, introduce leverage points, and cause heteroscedasticity ultimately degrading model accuracy. These effects underscore the importance of data transformations and robust model selection tailored to feature distribution properties.

Table 5 summarizes the performance of various models in predicting the features analyzed in Fig. 4, with the addition of thermal conductivity (THCD) at 1273 K, a critical property often used as a target or constraint in materials optimization. Results are presented as Mean Squared Error (MSE) alongside *R*-squared (R^2) values, reflecting each model's capacity to capture variance across features. XGBoost demonstrates strong overall performance, particularly for less complex features, but its accuracy declines for highly complex properties such as creep. Encoder–decoder models like FDN-R and DNF-net show comparable or superior performance on certain features, especially under higher complexity, while more expressive models such as 1D CNN exhibit inconsistent results, often struggling with extreme values. Notably, the VAE maintains stable predictive performance across increasing feature complexity, as evidenced in Fig. S4 by relatively aligned parity plots even for challenging targets like creep. This suggests that incorporating latent probabilistic representations can help the model capture distributions with heavy tails or multi-modality. Overall, while

Table 5 Test results on tabular dataset after hyper-parameter tuning. Presenting the performance for each model. MSE (R^2) are presented for five features

Model-feature name	YS 1000 C	1273 K density	1273 K THCD	Kou criteria	1300 °C creep CB
Skewness	0.48	0.77	0.97	2.2	39
Complexity	0.74	1.34	0.98	8.95	1560
XGBoost	1613 (R^2: 0.995)	0.141 (R^2: 0.920)	17.70 (R^2: 0.936)	2.61×10^4 (R^2: 0.854)	0.111 (R^2: -3.06×10^{21})
Encoder–decoder (DNF-net)	8949 (R^2: 0.976)	0.2755 (R^2: 0.902)	15.43 (R^2: 0.913)	2.59×10^4 (R^2: 0.855)	1.19×10^{-23} (R^2: 0.671)
Encoder–decoder (FDN-R)	5443 (R^2: 0.985)	0.2106 (R^2: 0.925)	14.56 (R^2: 0.918)	1.93×10^4 (R^2: 0.892)	8.32×10^{-24} (R^2: 0.771)
TabNet	1210 (R^2: 0.969)	0.2501 (R^2: 0.919)	13.73 (R^2: 0.917)	1.36×10^5 (R^2: 0.161)	2.58×10^{-23} (R^2: 0.403)
Encoder–decoder (1D CNN)	4348 (R^2: 0.881)	0.4662 (R^2: 0.833)	22.76 (R^2: 0.872)	4.48×10^5 (R^2: −1.5)	2.91×10^{-23} (R^2: 0.201)
Encoder–decoder (VAE)	1115 (R^2: 0.969)	0.2482 (R^2: 0.911)	19.53 (R^2: 0.889)	1.64×10^4 (R^2: 0.908)	1.14×10^{-23} (R^2: 0.683)

XGBoost is generally robust, the regularized FDN-R and the VAE show better adaptability to complex features, highlighting the importance of balancing model architecture with feature distribution characteristics.

4.4. Scaling and quantile effects on model performance

Fig. 5 presents a comparative analysis of the impact of various data scaling strategies on the predictive performance of the FDN-R model after just 50 training epochs. The parity plots show predictions on the original output scale, while the insets display model performance during training on the scaled data. For composite transformations (*e.g.*, NormalQuantile followed by MinMax scaling), inverse transformations were applied in reverse order to ensure predictions were accurately mapped back to the original scale. Without any scaling (Fig. 5a), the model fails to generalize, resulting in severely underestimated predictions and a negative R^2 value. This underscores the necessity of appropriate preprocessing, particularly when target distributions exhibit large variance or skewness.

Applying quantile transformations alone (Fig. 5b and d) improves training accuracy but fails to preserve this performance when outputs are inverse-transformed, resulting in a noticeable gap between predictions and the ground truth on the original scale. Although, uniform quantile transformation greatly improves the prediction accuracy. In contrast, combining normal quantile transformation with MinMax scaling (Fig. 5c) enhances model generalization across the output range which is comparable with uniform quintile transformation alone. This two-step normalization technique enables better distribution matching and numerical stability during training, which translates into more accurate predictions post-inversion.

Among the methods tested, MinMax scaling alone (Fig. 5e) yields the highest R^2 and lowest MSE on the original scale, indicating that for non-complex (moderately skewed or bounded) targets with narrow range, simpler transformations may suffice. The results suggest that normalization prior to training, especially when followed by an appropriate inverse transformation, plays an important role in ensuring the FDN-R model's robustness and generalization. MinMax and quantile-based strategies stand out as the most effective preprocessing methods, providing stable learning dynamics and superior performance in extrapolating complex material properties.

4.5. Suitability of models for predicting complex features

Several methods have been proposed to address the challenge of predicting highly skewed and complex properties like creep purely from a ML standpoint. First, more advanced pre-processing pipelines—such as adaptive or learned normalization layers—can help stabilize training on skewed targets. Fig. 5 highlights the critical impact of data pre-processing on the predictive performance. Second, loss functions tailored to emphasize performance on underrepresented or extreme values (*e.g.*, quantile loss, SMAPE, or hybrid losses) can guide models to focus on the tail behavior of the distribution. Third, model architectures that are robust to non-Gaussian distributions, such as attention-based models (*e.g.*, TabNet) or uncertainty-aware frameworks (*e.g.*, Bayesian neural networks), can improve generalization. Lastly, data augmentation techniques—such as synthetic minority oversampling or generative modeling—can help rebalance skewed distributions and reduce overfitting. Together, these approaches offer a practical path to improve prediction of complex features even in the absence of additional physics-based inputs.

In addition to advanced pre-processing pipelines, we also explored TabNet, an attention-based model for tabular data. As shown in the second row from the bottom in Fig. S4, TabNet achieves competitive performance for moderately complex features such as density and thermal conductivity but begins to show limitations with the Creep CB, the most skewed feature. While its use of attentive feature selection allows it to prioritize important signals and maintain relatively strong predictions on moderately non-Gaussian distributions, its performance deteriorates on severely skewed data, consistent with models that lack explicit handling of tail behavior. This suggests that while TabNet's inductive bias is useful for structured data, additional architectural or training modifications are necessary for generalizing to extreme outputs like creep. Overall, the results underscore the importance of aligning model choice with feature complexity.

4.6. Generalization of models to other datasets

To assess the robustness and portability of deep tabular models beyond a single dataset, we evaluated their generalization performance on two additional datasets: BIRDSHOT and MPEA. These datasets differ significantly from ATLAS-RHEA in terms of size, data sparsity, property types, and statistical distributions, providing a diverse testing ground for model adaptability.

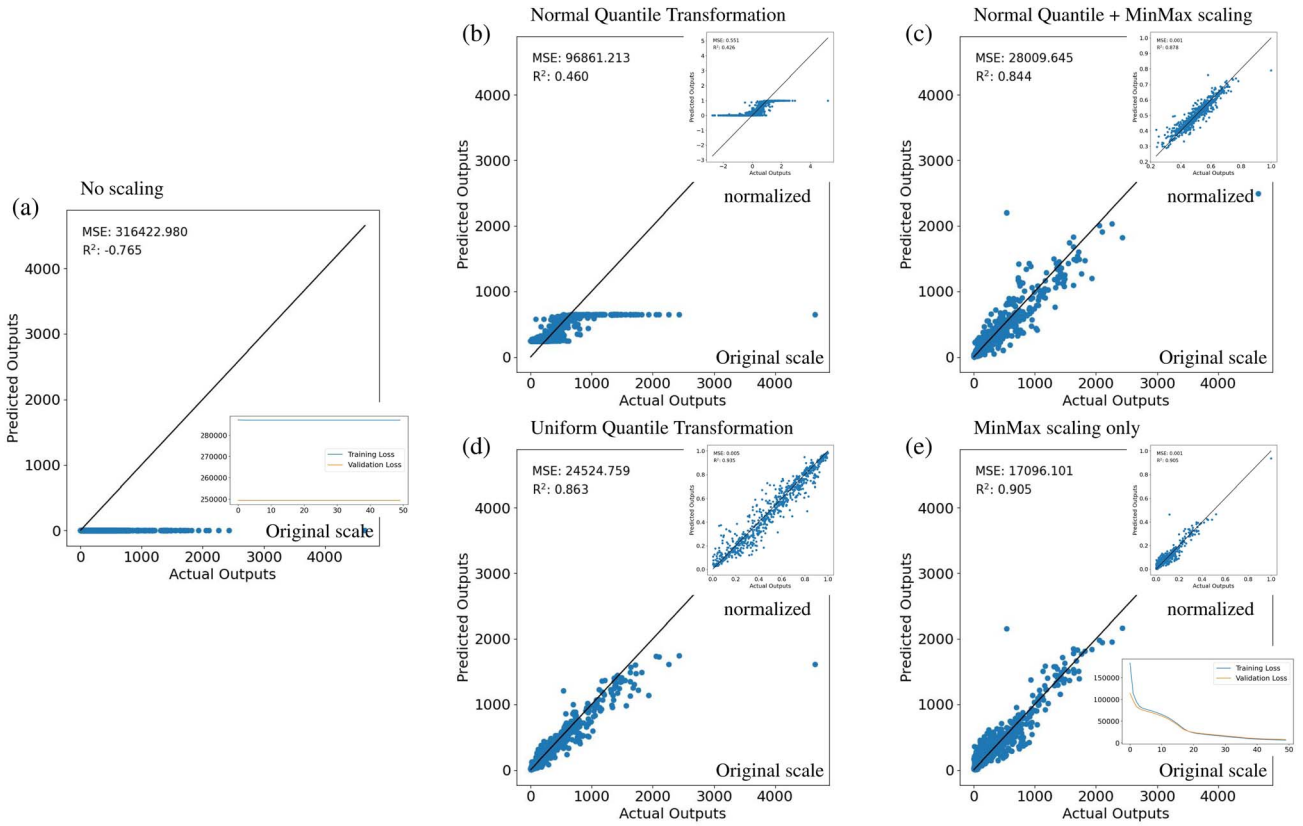


Fig. 5 Parity plots showing the effect of different scaling methods on the FDN-R model's predictive performance (test data) after 50 epochs for single-task prediction of the Kou criteria feature. (a) No scaling, (b) normal quantile transformation, (c) normal quantile followed by MinMax scaling, (d) uniform quantile transformation, and (e) MinMax scaling only. The smaller insets show training performance on the scaled data. Applying normalization followed by appropriate inverse transformations leads to substantial improvements in accuracy, with MinMax and quantile-based methods yielding the best generalization on the original scale. Inset plots specifically show predictions on data scaled *via* sequential NormalQuantile and MinMax transformations, then accurately mapped back to the physical scale using inverse operations.

In the BIRDSHOT dataset, which focuses on non-equimolar Cantor HEAs with moderately skewed mechanical properties (*e.g.*, yield strength, elongation, hardness), both XGBoost and the regularized FDN-R maintained strong predictive performance. However, the DNF-Net demonstrated superior resilience to moderate outliers and class imbalance in features such as hardness (skewness = 2.5, kurtosis = 10), outperforming XGBoost in terms of both R^2 and SMAPE. For the MPEA dataset, where feature completeness and measurement consistency varied across properties, encoder–decoder models, particularly the regularized FDN-R, showed strong generalization despite the absence of hyperparameter tuning. Without task-specific optimization, the FDN-R model produced high-quality predictions for key properties such as ultimate tensile strength and calculated density, with parity plots confirming alignment between predicted and observed values.

The generalization trends indicate that model architecture must be matched not only to the statistical characteristics of the target feature but also to the broader dataset conditions such as sample size, missingness, and feature granularity. The overcomplete encoder–decoder models with Regularized FDN-R and DNF-Net architectures offer a promising balance between expressiveness and robustness, while simpler models like XGBoost remain effective baselines in small or sparse datasets.

4.7. Graphical interface for data-driven materials modeling

To support reproducible, efficient, and user-friendly deployment of the encoder–decoder workflows explored in this study, we developed an interactive graphical user interface (GUI) (Fig. 6). The application integrates all essential steps, from dataset upload, feature selection, and preprocessing, including thresholding and advanced scaling transformations, to customizable neural network architecture definition, training execution, and results visualization. This allows researchers to iteratively explore how preprocessing choices and hyperparameter configurations affect model generalization, offering immediate insights into complex feature–property relationships within tabular materials data.

Beyond accelerating exploratory research, the GUI serves as an effective educational platform. It enables graduate students and interdisciplinary collaborators to experiment with data transformations, model architectures, and training regimes in a transparent, hands-on manner, reinforcing foundational machine learning concepts without requiring extensive programming expertise. This lowers the barrier for adoption of advanced encoder–decoder techniques in materials science, supporting a broader community of practitioners aiming to leverage data-driven strategies.

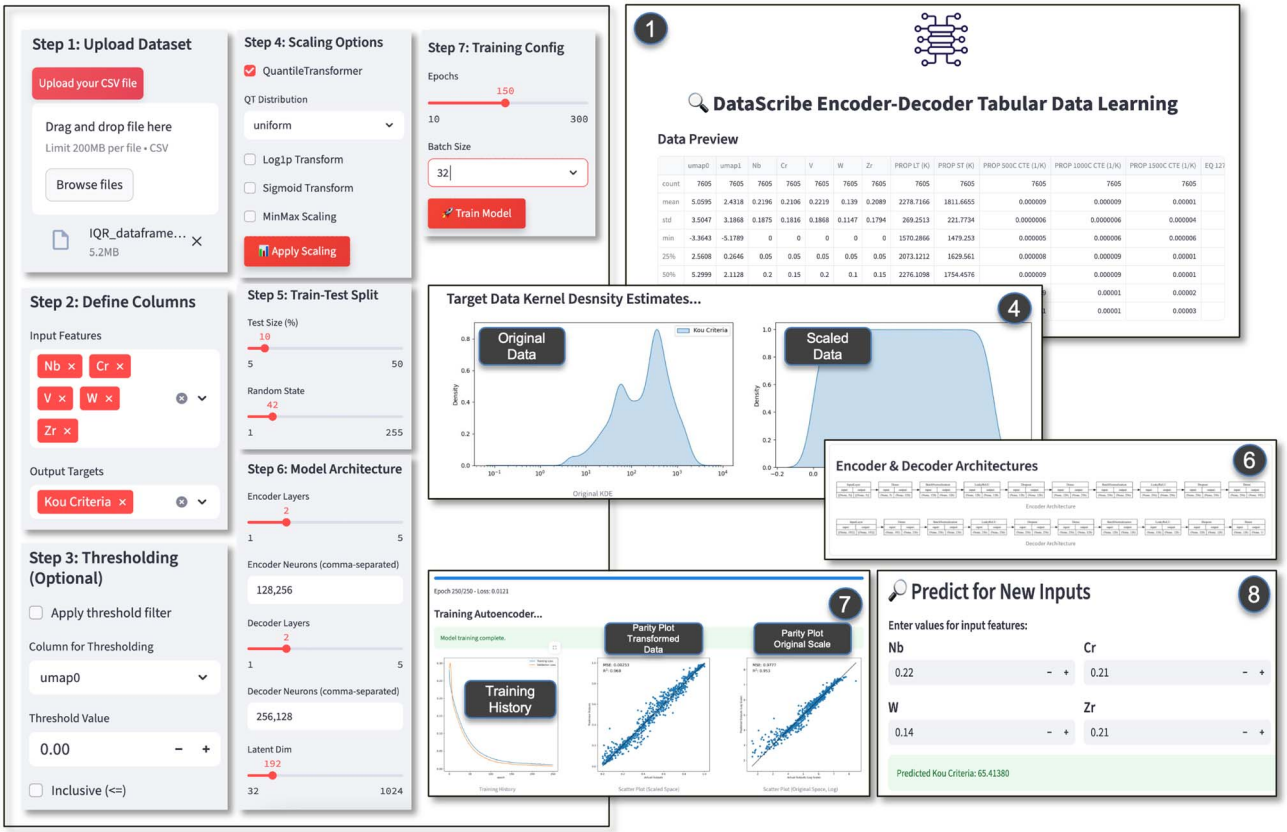


Fig. 6 Overview of the GUI application developed for interactive encoder–decoder learning on tabular materials data. Left-hand side: sequential application sections guiding the user through dataset upload, feature and target selection, optional thresholding, scaling transformations, train-test split configuration, and neural network architecture specification. Right-hand side: outputs from the application including the dataset preview (step 1), kernel density estimates comparing original and scaled target distributions (step 4), automatically generated encoder–decoder architecture diagrams (step 6), training history and parity plots (step 7), and an interface for predicting target properties from user-defined inputs (step 8).

Importantly, the software is designed to be modular and extensible, providing a foundation for future integration of uncertainty quantification, multi-objective optimization, and autonomous decision-making workflows. By openly sharing this tool, we aim to promote collaborative exploration and accelerate the application of robust machine learning approaches to the discovery and design of advanced materials systems.

5. Conclusion

This study systematically evaluated the use of encoder–decoder model for data transformation using FDN-R, DNF-Net, 1D CNN, TabNet, and VAE architectures, against the benchmark XGBoost model for tabular materials data. While tree-based methods like XGBoost maintained competitive performance and fast convergence across balanced features, encoder–decoder models, particularly using DNF-Net and regularized FDN-R architectures, showed improved generalization to highly skewed and non-linear properties such as creep resistance.

Our results indicate that these neural architectures can rival and, in some cases, surpass traditional models in accuracy, particularly for complex distributions, provided that appropriate data scaling and regularization techniques are applied. Regularized FDN-R models showed robust generalization across varying feature complexities, while DNF-Nets offered a favorable trade-off between interpretability and performance. Conversely, models with higher capacity, such as CNNs, were more sensitive to outliers and skewed data, highlighting the importance of matching architecture complexity to feature distribution.

Beyond predictive tasks, encoder–decoder frameworks offer exciting potential for downstream applications in materials discovery pipelines, including real-time optimization, synthetic data generation for privacy preservation, and closed-loop predictions acting as machine learning priors in Bayesian discovery of materials. Future progress will depend on addressing challenges around convergence stability, uncertainty quantification, and computational efficiency for small datasets.

In summary, while no single model universally outperforms across all scenarios, hybrid and interpretable deep learning approaches, when thoughtfully applied, can enhance the accuracy, adaptability, and utility of machine learning in materials science, particularly for complex applications.

Conflicts of interest

There are no conflicts to declare.

Data availability

The data used as part of this study are publicly available at GitHub (**https://github.com/vahid2364/DataScribe_DeepTabularLearning.git**), Zenodo (**https://doi.org/10.5281/zenodo.16396374**), and DataScribe platform (**https://datascribe.cloud**). The BIRDSHOT dataset is actively updated and new alloys are being added as part of ongoing Bayesian optimization campaigns, and the authors encourage researchers to access the latest version and contribute to its development through collaborative data sharing. The developed models are available on the DataScribe Online Platform subject to certain conditions. Access to the trained models *via* the DataScribe Online Platform may be subject to conditions such as user registration, non-commercial use, proper citation, or agreement to a data usage policy. Interested users should refer to the platform's access terms for details.

The Supplementary Information includes additional details on model architectures (DNF-Net, TabNet, and VAE), training procedures, hyperparameter configurations, and evaluation metrics used in this study. It also provides extended results for additional datasets (BIRDSHOT and MPEA), and supplementary figures and tables supporting the main findings. Supplementary information is available. See DOI: **https://doi.org/10.1039/d5dd00166h**.

Acknowledgements

The authors acknowledge the Grace supercomputing facility at Texas A&M University for providing essential computational resources that facilitated the research presented in this paper. VA and RA also acknowledge the financial support from the BIRDSHOT Center, provided under Cooperative Agreement Number W911NF-22-2-0106.

References

1 T. N. Kipf, and M. Welling, Semi-supervised classification with graph convolutional networks, *arXiv*, 2016, preprint, arXiv:1609.02907, DOI: **10.48550/arXiv.1609.02907**.
2 S. R. Kalidindi and M. De Graef, Materials informatics: Analyzing and accelerating the discovery of engineering materials, *Annu. Rev. Mater. Res.*, 2015, **45**, 171–193.
3 D. Raabe, C. C. Tasan and E. A. Olivetti, Strategies for improving the sustainability of structural metals, *Nature*, 2019, **575**(7781), 64–74.
4 K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev and A. Walsh, Machine learning for molecular and materials science, *Nature*, 2018, **559**(7715), 547–555.
5 V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk and G. Kasneci, Deep neural networks and tabular data: A survey, *IEEE Transact. Neural Networks Learn. Syst.*, 2022, 7499–7519.
6 T. Xie and J. C. Grossman, Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties, *Phys. Rev. Lett.*, 2018, **120**(14), 145301.
7 R. Chen, S. P. Ong and J. L. Kuo, Atomsets as a hierarchical transfer learning framework for small and large materials datasets, *npj Comput. Mater.*, 2019, **5**(1), 103.
8 D. Jha, L. Ward, A. Paul, W.-k. Liao, A. Choudhary, C. Wolverton and A. Agrawal, Elemnet: Deep learning the chemistry of materials from only elemental composition, *Sci. Rep.*, 2018, **8**(1), 17593.
9 A. Agrawal and A. Choudhary, Deep materials informatics: Applications of deep learning in materials science, *MRS Commun.*, 2019, **9**(3), 779–792.
10 E. D. Cubuk, S. S. Schoenholz, J. M. Rieser, *et al.*, Identifying structural flow defects in disordered solids using machine-learning methods, *Phys. Rev. Lett.*, 2015, **114**(10), 108001.
11 S. M. Tazwar, M. Knobbout, E. H. Quesada, and M. Popa, Tab-vae: A novel VAE for generating synthetic tabular data, in *ICPRAM*, 2024, pp. 17–26.
12 L. Xu, K. Veeramachaneni, Synthesizing tabular data using generative adversarial networks, *arXiv*, 2018, preprint, arXiv:1811.11264, DOI: **10.48550/arXiv.1811.11264**.
13 K. R. Das and A. Imon, A brief review of tests for normality, *Am. J. Theor. Appl. Stat.*, 2016, **5**(1), 5–12.
14 J. Shi, D. Luo, X. Wan, Y. Liu, J. Liu, Z. Bian and T. Tong, Detecting the skewness of data from the five-number summary and its application in meta-analysis, *Stat. Methods Med. Res.*, 2023, **32**(7), 1338–1360.
15 P. Linardatos, V. Papastefanopoulos and S. Kotsiantis, Explainable ai: A review of machine learning interpretability methods, *Entropy*, 2020, **23**(1), 18.
16 T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden and A. Peste, Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks, *J. Mach. Learn. Res.*, 2021, **22**(241), 1–124.
17 S. Popov, S. Morozov, and A. Babenko, Neural oblivious decision ensembles for deep learning on tabular data, *arXiv*, 2019, preprint, arXiv:1909.06312, DOI: **10.48550/arXiv.1909.06312**.
18 G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, Lightgbm: A highly efficient gradient boosting decision tree, in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 3146–3154.
19 L. Fiedler, T. Schulte and H. Hirschfeld, Tabular neural networks for modeling real-world systems, *J. Artif. Intell. Res.*, 2021, **72**, 1–22.
20 V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, and G. Kasneci, Deep neural networks and tabular data: A survey, *arXiv*, 2021, preprint, arXiv:2110.01889, DOI: **10.48550/arXiv.2110.01889**.
21 G. Somepalli, M. Goldblum, A. Schwarzschild, J. Geiping, and T. Goldstein, Saint: Improved neural networks for tabular data *via* row attention and contrastive pre-training, in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021, vol. 34, pp. 29941–29953.

22 S. Ö. Arik, and T. Pfister, Tabnet: Attentive interpretable tabular learning, in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, vol. 35, pp. 6679–6687.
23 Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko, Revisiting deep learning models for tabular data, in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021, vol. 34, pp. 18932–18943.
24 X. Huang, A. Khetan, M. Cvitkovic, and Z. Karnin, Tabtransformer: Tabular data modeling using contextual embeddings, in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020, vol. 33, pp. 14943–14953.
25 G. Ke, Q. Liu, T. Wang, W. Chen, X. Ma, and Q. Ye, Deepgbm: A deep learning framework distilled by gbdt for online prediction tasks, in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 384–394.
26 H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, *et al.*, Wide & deep learning for recommender systems, in *Proceedings of the 1st workshop on deep learning for recommender systems*, 2016, pp. 7–10.
27 L. Katzir, G. Elidan, and R. El-Yaniv, *Net-dnf: Effective deep modeling of tabular data, in International conference on learning representations*, 2020.
28 F. Doshi-Velez, and B. Kim, Towards a rigorous science of interpretable machine learning, *arXiv*, 2017, preprint, arXiv:1702.08608, DOI: **10.48550/arXiv.1702.08608**.
29 M. T. Ribeiro, S. Singh, and C. Guestrin, "why should i trust you?" explaining the predictions of any classifier, in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.
30 L. Balderas, M. Lastra and J. M. Benítez, Optimizing dense feed-forward neural networks, *Neural Netw.*, 2024, **171**, 229–241.
31 E. Barnes and J. Hutson, Architectural elements contributing to interpretability of deep neural networks (dnns), *International Journal of Multidisciplinary and Current Educational Research*, 2024, **6**(3), 218–225.
32 S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj and D. J. Inman, 1d convolutional neural networks and applications: A survey, *Mech. Syst. Signal Process.*, 2021, **151**, 107398.
33 A. J. Ferreira, and M. A. Figueiredo, *Boosting algorithms: A review of methods, theory, and applications, Ensemble machine learning: Methods and applications*, 2012, pp. 35–85.
34 Y. Chen, A. Huang, Z. Wang, I. Antonoglou, J. Schrittwieser, D. Silver, and N. de Freitas, Bayesian optimization in alphago, *arXiv*, 2018, preprint, arXiv:1812.06855, DOI: **10.48550/arXiv.1812.06855**.
35 J. Bergstra, R. Bardenet, Y. Bengio and B. Kégl, Algorithms for hyper-parameter optimization, *Advances in Neural Information Processing Systems*, 2011, vol. 24.
36 T. Yu, and H. Zhu, Hyper-parameter optimization: A review of algorithms and applications, *arXiv*, 2020, preprint, arXiv:2003.05689, DOI: **10.48550/arXiv.2003.05689**.
37 J. Bergstra, D. Yamins, and D. Cox, Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures, in *International Conference on Machine Learning*, PMLR, 2013, pp. 115–123.
38 T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, Optuna: A next-generation hyperparameter optimization framework, in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.
39 M. D. Allen, V. Attari, B. Vela, J. Hanagan, R. Malak, and R. Arróyave, Performance-driven computational design of multi-terminal compositionally graded alloy structures using graphs, *arXiv*, 2024, preprint, arXiv:2412.03674, DOI: **10.48550/arXiv.2412.03674**.
40 M. Mulukutla, A. N. Person, S. Voigt, L. Kuettner, B. Kappes, D. Khatamsaz, R. Robinson, D. S. Mula, W. Xu, D. Lewis, *et al.*, Illustrating an effective workflow for accelerated materials discovery, *Integrating Materials and Manufacturing Innovation*, 2024, pp. 1–21.
41 C. K. Borg, C. Frey, J. Moh, T. M. Pollock, S. Gorsse, D. B. Miracle, O. N. Senkov, B. Meredig and J. E. Saal, Expanded dataset of mechanical properties and observed phases of multi-principal element alloys, *Sci. Data*, 2020, **7**(1), 430.