



ARL-TR-10473 • OCT 2026



Explainable Machine Learning for Predicting Steel Toughness Using Multitask Sparse Neural Networks and CALPHAD Augmentation

by B. Christopher Rinderspacher, Benjamin Szajewski,
Heather Murdoch, and Gregory Barber

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.



Explainable Machine Learning for Predicting Steel Toughness Using Multitask Sparse Neural Networks and CALPHAD Augmentation

**B. Christopher Rinderspacher, Benjamin Szajewski, Heather Murdoch,
and Gregory Barber**

DEVCOM Army Research Laboratory

REPORT DOCUMENTATION PAGE

1. REPORT DATE		2. REPORT TYPE		3. DATES COVERED					
October 2026		Technical Report		<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">START DATE</td> <td style="width: 50%;">END DATE</td> </tr> <tr> <td>1 October 2024</td> <td>2 September 2026</td> </tr> </table>		START DATE	END DATE	1 October 2024	2 September 2026
START DATE	END DATE								
1 October 2024	2 September 2026								
4. TITLE AND SUBTITLE									
Explainable Machine Learning for Predicting Steel Toughness Using Multitask Sparse Neural Networks and CALPHAD Augmentation									
5a. CONTRACT NUMBER		5b. GRANT NUMBER		5c. PROGRAM ELEMENT NUMBER					
5d. PROJECT NUMBER		5e. TASK NUMBER		5f. WORK UNIT NUMBER					
6. AUTHOR(S)									
B. Christopher Rinderspacher, Benjamin Szajewski, Heather Murdoch, and Gregory Barber									
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)				8. PERFORMING ORGANIZATION REPORT NUMBER					
DEVCOM Army Research Laboratory ATTN: FCDD-RLA-MC Adelphi, MD 20783				ARL-TR-10473					
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)			10. SPONSOR/MONITOR'S ACRONYM(S)		11. SPONSOR/MONITOR'S REPORT NUMBER(S)				
12. DISTRIBUTION/AVAILABILITY STATEMENT									
DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.									
13. SUPPLEMENTARY NOTES									
ORCIDs: B. Christopher Rinderspacher, 0000-0003-1333-1653; Benjamin Szajewski, 0000-0002-5028-414X; Heather Murdoch, 0000-0001-7710-0577; Gregory Barber, 0000-0002-6919-5083									
14. ABSTRACT									
<p>Neural networks model complex relationships found in data, including uncertainty. However, they traditionally function as “black boxes,” without providing mechanistic explanations for their predictions. This report presents a systematic evaluation of a novel method for training fully connected neural networks based on information-theoretic regularizations. These regularizations are designed to both sparsify the network structure and provide mechanistic insights into predictive computations.</p> <p>We applied this methodology to predict the Charpy V-Notch (CVN) and Izod impact toughness of steels. We compared models trained on composition-only inputs (20 variables) against those trained on inputs augmented with thermodynamic properties computed from CALPHAD (24 variables). We investigated the impact of three parameter p scheduling regimes—LinearScheduler, PiecewiseScheduler, and ExpScheduler—against a constant p baseline (no schedule).</p> <p>The empirical results show that enforcing information-theoretic regularizations does not degrade neural network performance; rather, when paired with appropriate schedules and CALPHAD augmentations, it yields highly sparsified networks with superior generalization. Specifically, the PiecewiseScheduler achieved the highest average network sparsification ($\approx 84.36\%$ pruned CVN sparsity, $\approx 83.81\%$ pruned Izod sparsity), while the LinearScheduler combined with CALPHAD-augmented inputs produced the most accurate models, reaching a validation loss of 0.31 and high post-pruning R^2 values ($R^2_{CVN} = 0.73$, $R^2_{IZOD} = 0.83$).</p>									
15. SUBJECT TERMS									
Sciences of Extreme Materials, explainable machine learning, explainable ML, multitask networks, Charpy V-Notch, CVN, IZOD, optimization									
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT		18. NUMBER OF PAGES				
a. REPORT	b. ABSTRACT	c. THIS PAGE	UU		32				
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED							
19a. NAME OF RESPONSIBLE PERSON				19b. PHONE NUMBER (Include area code)					
B. Christopher Rinderspacher				(520) 691-6019					

STANDARD FORM 298 (REV. 5/2020)

Prescribed by ANSI Std. Z39.18

Contents

List of Figures	v
List of Tables	v
Declaration of Generative AI and AI-Assisted Technologies	vi
1. Introduction	1
2. Methods	3
2.1 Network Architecture and Primary Loss Function	3
2.2 Information-Theoretic and Activity Regularization	3
2.3 Bilevel Multi-Objective Optimization	4
2.4 Experimental Design and p Schedules	4
2.5 Dataset	5
3. Results and Analytical Discussion	5
3.1 Performance Comparison across Parameter p Schedules	5
3.2 Impact of Physical Augmentation (CALPHAD vs. Composition-Only)	6
3.3 Effect of Hyperparameters (Learning Rate and Batch Size)	7
3.4 Analysis of Outstanding Runs	7
3.5 Pareto Front Visualization and Input Augmentation Analysis	8
3.6 Analytical Evaluation of Bilevel Monotonicity	8
3.7 Stochastic Dynamics and Variance Control	9
3.8 Staircase vs. Mathematically Continuous Schedules	10
3.9 Homotopy, Continuation, and Pruning Threshold Dynamics	10
4. Significance of the Differences: Standard Deviation and Generalization Stability	11
4.1 Statistical Robustness of CALPHAD-Augmented vs. Composition-Only Models	11
4.2 Within-Run Epoch Stochasticity	12

4.3	Statistical Equivalence Bins	12
4.4	Analytical Evaluation of Pruning Stability Constraints	14
4.5	Replicate Stability and LWIOI Dominance	15
5.	Future Work	16
5.1	Tuning Optimizer Momentum	16
5.2	Statistical Robustness and Global Optimization	16
6.	Conclusions and Recommendations	16
7.	References	18
	Appendix. Python Data Analysis Script	20
	List of Symbols, Abbreviations, and Acronyms	23
	Distribution List	24

List of Figures

- Figure 1. Pareto Front of Pruned CVN Sparsity vs. Final Validation Loss. This figure plots the optimal trade-off boundaries (Pareto frontiers) between network sparsity (the percentage of pruned weights) and generalization capability (validation loss) for the two input regimes. The orange curve tracks the CALPHAD-augmented models, while the purple curve tracks the composition-only models. Incorporating CALPHAD properties shifts the entire optimal front downward, achieving a lower validation loss floor across all sparsity levels and preventing the performance collapse observed in composition-only models at extreme compression ($>83\%$). 8
- Figure 2. Neural network connection graph of 1ffe9bb2 for Izod prediction. Width of arrows indicates magnitude of weight. Blue arrows are negative and red arrows are positive. Hue of nodes reflects average magnitude of outgoing weights. Sparsity is $\approx 80.00\%$ 14

List of Tables

- Table 1. Performance comparison by p -schedule. 6
- Table 2. Performance comparison by dataset type. 6
- Table 3. Top five performers with modeling parameters and performance metrics..... 7
- Table 4. Binning of runs by statistical significance ($p > 0.05$). 13

Declaration of Generative AI and AI-Assisted Technologies

Generative AI (GenAI, Gemini 3.5 Flash) was used to generate the initial draft and code snippets for analysis and visualization. Code was line-reviewed and modified to meet intent and correctness. All references were checked and discarded if inappropriate. The text was reviewed and modified repeatedly by all authors for correctness and intent, adding missing arguments, data, or references.

1. Introduction

Fully Connected Neural Networks (FCNs) are powerful universal function approximators capable of representing complex, nonlinear relationships across high-dimensional features, as mathematically established by Cybenko (1989) for sigmoidal activations and expanded by Hornik et al. (1989) for arbitrary feedforward architectures. In materials science, deep learning has been successfully deployed to predict complex mechanical properties (Agrawal and Choudhary 2016), such as impact toughness, where chemical composition and microstructural features interact nonlinearly to govern macroscopic performance (Bhadeshia 1999, 2009; Sha and Edwards 2007).

However, in order to usefully apply a model to material and process design with confidence in the property predictions, the model must be explainable. The expressivity of FCNs is vast and degenerate. As discussed by Goodfellow et al. (2016), the non-convex optimization landscapes of deep networks contain infinitely many weight configurations (saddle points and local minima) that achieve identical minimal loss on a training set. Because standard FCNs function as opaque “black boxes,” they run the risk of exploiting spurious statistical correlations in composition noise rather than learning the true underlying physics.

Explainability also depends on comprehensive training datasets that encompasses the desired input feature space and underlying physical mechanisms. Specifically for impact toughness in steels, both the cantilever Izod impact test and simply supported Charpy V-Notch (CVN) testing are standardized (ASTM International 2018) but there is no conversion between the two methods, creating challenges in assembling a large, robust dataset. In the mid-20th century, the demonstrated necessity of determining the ductile-to-brittle transition temperature (DBTT) in steels supported the use of the CVN test (Tóth et al. 2002). The Izod configuration clamps a notched specimen vertically as a cantilever beam in a heavy metallic vice while the CVN test supports a notched specimen horizontally and strikes it in a three-point bending configuration (Dieter 1986). Mapping the DBTT requires testing steel specimens across a broad range of temperatures including sub-ambient. In the Izod setup, clamping the pre-cooled specimen in the vice is time-sensitive as the sample may warm too much before the pendulum can strike. By contrast, the Charpy fixturing supports a fast transition from cooling medium to test ensuring more accurate low-temperature measurements.

Many sources of historical alloy data consist almost exclusively of Izod impact energies, whereas more recent datasets contain CVN toughness values. To bridge this database gap, Multitask Learning (MTL) networks may be deployed. Multitask

deep neural networks utilize shared representation layers to predict multiple correlated tasks simultaneously (Caruana 1997). Training a single (MTL) network to predict both Izod and CVN values simultaneously forces the hidden layers to learn a unified, low-dimensional representation of the steel's toughness, which implicitly provides an impact toughness conversion. Through this inductive transfer, the abundant Izod database actively regularizes and improves the generalization of the scarcer CVN datasets, resolving task-specific noise and guiding the optimizer away from degenerate local minima (Ramsundar et al. 2015).

To navigate the bias-variance tradeoff in FCNs and restrict parameter degeneracy, conventional regularization functions are traditionally utilized. For generalization and weight shrinkage, ℓ_2 -regularization—originating in statistics as ridge regression by Hoerl and Kennard (1970) and formalized as weight decay in neural networks by Krogh and Hertz (1992)—penalizes the Euclidean norm of the parameters, preventing individual weights from exploding. To induce sparsity, ℓ_1 -regularization—introduced as the Least Absolute Shrinkage and Selection Operator by Tibshirani (1996)—penalizes the absolute sum of the weights, driving nonessential weights to exact zero to perform automatic feature selection.

While ℓ_1 - and ℓ_2 -regularizers manage overfitting, they operate blindly on parameter magnitudes without considering the physical or information-theoretic flow within the network. In an MTL network, this blind shrinkage fails to resolve the conversion interpretability bottleneck, leaving the Izod and CVN representations mixed in an untraceable parameter web.

This study directly addresses this bottleneck by implementing a differentiable, information-theoretic regularization framework that goes beyond simple weight magnitude shrinkage to directly minimize mutual information and activation correlation between nodes during training.

This network sparsification approach builds conceptually on historical pruning frameworks like Optimal Brain Damage (OBD) by LeCun et al. (1990) and Optimal Brain Surgeon (OBS) by Hassibi and Stork (1993). While OBD and OBS prune trained dense networks post-hoc using second-order Taylor approximations of the Hessian matrix, modern explanatory networks—such as Self-Explaining Neural Networks (Alvarez-Melis and Jaakkola 2018) and Neural Additive Models (Agrawal et al. 2021)—embed interpretability directly into the optimization objective during training.

2. Methods

2.1 Network Architecture and Primary Loss Function

We train a multitask FCN consisting of 4 hidden layers with 48 nodes each to predict the mean (μ_i) and variance (σ_i^2) of both CVN and Izod toughness. The network uses the rectified linear unit (ReLU) activation function. To capture heteroscedastic uncertainty, we define the primary validation/training loss $L(x, y)$ as the power of a Gaussian negative log-likelihood:

$$L(x, y) = \left(\left(\frac{1}{n} \right) \sum_i \left(-\frac{(\mu_i - y_i)^2}{2\sigma_i^2} - \ln \sigma_i \right)^6 \right)^{\frac{1}{4}}, \quad (1)$$

where n is the total number of observations, y_i is the i -th experimental observation, μ_i is the corresponding predicted mean, and σ_i^2 represents the corresponding predicted variance.

2.2 Information-Theoretic and Activity Regularization

To isolate and separate the compositional information processed by the network, we minimize the mutual information between separate nodes within the same layer. The mutual information is minimized if their supports do not overlap. The graph $\{(f(v^T x), f(w^T x)) | x \in X\}$ exhibits minimal mutual information if $v^T w = 0$. Enforcing the stricter condition $\forall_i w_i v_i = 0$ separates information into individual entries.

We propose the differentiable layer-wise weight regularizer $\hat{O}(W; p)$ to penalize overlapping node ranges:

$$\hat{O}(W; p) = \|\log^\odot 1 + p \text{abs}^\odot(W)\|_2^2 - \|(\log^\odot 1 + p \text{abs}^\odot(W))^\odot\|_1^2, \quad (2)$$

where $\log^\odot$ and $\text{abs}^\odot$ represent element-wise application of the logarithm and absolute value functions, respectively, and W is the weight matrix of a layer. As the tuning parameter $p \rightarrow \infty$, $\hat{O}(W; p)$ smoothly approaches the discrete ℓ_0 -norm (the count of nonzero elements). This regularizer will henceforth be denoted in this report as the Layer-Wise Information Overlap Inhibitor (LWIOI). An alternate formulation $\hat{O}(W^T; p)$ is also implemented to normalize layer dimension disparities and reduce scaling degeneracies, denoted as the Layer-Wise Information Mixing Inhibitor (LWIMI).

To penalize the correlation of activations directly, we employ the activity metric $\hat{A}(a; p)$ across activation vectors a^s for samples $s \in [S]$:

$$\hat{A}(a; p) = \frac{1}{S} \sum_{s \in [S]} \sum_{i=1, j=1}^n \frac{\log 1 + pa_i^s a_j^s}{\log 1 + p} - \left(\frac{1}{S} \sum_{s=1}^S \sum_{i=1}^n \frac{\log 1 + pa_i^s}{\log 1 + p} \right)^2, \quad (3)$$

Because the network utilizes ReLU activations, $\hat{A}(a; p)$ mathematically approaches the joint probability of nonzero joint activations as $p \rightarrow \infty$.

2.3 Bilevel Multi-Objective Optimization

The network parameters are optimized under a bilevel scheme:

$$\begin{aligned} & \min_{W, \alpha} L_v(W) \\ \text{s. t. } & W \in \arg \min_{\tilde{W}} \sum_i \alpha_i P_i(\tilde{W}) \end{aligned} \quad (4)$$

where P_i represents the individual layer regularizers and primary training losses, and L_v represents the validation loss. The trade-off vector α is solved dynamically at each step by projecting the validation loss gradient onto the regularizer gradients via a quadratic program (QP):

$$\begin{aligned} \alpha &= \arg \min_{\tilde{\alpha} \geq 0} \left\| \nabla L_v - \sum_i \tilde{\alpha}_i \nabla P_i \right\|^2 \\ \text{s. t. } & 1^T \tilde{\alpha} = 1 \end{aligned} \quad (5)$$

The model weights are updated to minimize $\sum_i \alpha_i P_i$.

We use mini-batch stochastic gradient descent with multiple learning rate schedules and batch sizes at each optimization step (Bottou et al. 2016). Since the validation set is smaller than the training set, the validation set is repeated and shuffled to match the number of batches in the training set each epoch. The training set is shuffled each epoch as well. Each step in an epoch evaluates one training batch and one validation batch.

2.4 Experimental Design and p Schedules

The parameter p governs the strictness of the ℓ_0 -norm approximation. If p is kept constant, the network struggles to navigate the non-convex landscape. Therefore, we evaluate three scheduling policies to dynamically increase p during training:

- **LinearScheduler:** Linearly increases p from an initial value V to a target value at a constant rate.
- **PiecewiseScheduler:** Modulates p in step-wise phases (specifically a continuous cosine phase followed by a constant flat phase).
- **ExpScheduler:** Geometrically/exponentially escalates p .

- None (Constant p): Acts as the baseline, holding p constant throughout the 120,001 training epochs.

2.5 Dataset

The dataset comprises 2910 entries of composition, processing, augmented features, and target property for martensitic steels.

Input features consist of 17 elements for the composition of the steel (C, Mn, Si, Cr, Ni, Mo, V, Co, Al, W, Cu, Nb, Ti, B, N, P, S) and 3 processing conditions (austenitization temperature, tempering time, and temperature). Additionally, four properties computed via CALPHAD (CALculation of PHase Diagrams) were added, considered “augmented features.” The CALPHAD methodology uses thermodynamic data to calculate phase equilibria as a function of composition and temperature. The four features used here were determined by steel metallurgists as strong contenders for correlation with the following target properties: martensite start temperature calculated using the Ghosh–Olsen model (M_s Ghosh) as per Ghosh and Olson (1994, 2001), transformation temperatures calculated using a custom Python algorithm and Thermo-Calc software (A_{c1} , A_{c3}), and the volume fraction of non-austenitic phases at the austenitization temperature, calculated using Thermo-Calc software (vf). The dataset included 630 entries with CVN values and 2308 entries with Izod values, where 14 entries provide both CVN and Izod values). Strictly positive features, excluding composition inputs, are transformed by the natural logarithm. All inputs are subsequently normalized to a zero center, unit Gaussian.

The training set and validation set are derived by stratifying CVN and Izod, respectively, and randomly assigning each stratum into two-thirds training set and one-third validation set.

3. Results and Analytical Discussion

The Appendix provides scripts that were used to analyze the results and provide the graphs and plots.

3.1 Performance Comparison across Parameter p Schedules

Table 1 compiles the mean and standard deviation of predictive and topological performance metrics across all 47 successfully completed optimization runs, grouped by their parameter p scheduling policy.

Table 1. Performance comparison by p -schedule.

Schedule	Count	Pruned CVN R^2 (Mean $\pm$ SD)	Pruned IZOD R^2 (Mean $\pm$ SD)	Val. loss (Mean $\pm$ SD)	Avg. non-zero weights	Pruned sparsity
ExpScheduler	1	0.630	0.750	0.363	2194.0	72.95
LinearScheduler	34	0.569 ± 0.443	0.753 ± 0.057	0.392 ± 0.055	2589.6 ± 1917.6	68.20
None (Constant $p = 1$)	9	0.553 ± 0.116	0.684 ± 0.110	0.497 ± 0.151	6979.0 ± 81.8	13.89
PiecewiseScheduler	3	0.650 ± 0.010	0.757 ± 0.012	0.402 ± 0.010	1313.7 ± 142.5	84.36

The constant p baseline fails to sparsify the network, yielding a low average pruned sparsity of 13.89% (retaining an average of 6,979 active weights out of 8,112). It also results in the worst generalization, with an average validation loss of 0.497.

The PiecewiseScheduler provides the highest structural sparsification, pruning 84.36% of all network weights (retaining as low as 1,314 active weights). Despite this high compression, it maintains strong average predictive performance ($R^2_{\text{CVN}} = 0.650$, $R^2_{\text{IZOD}} = 0.757$, and validation loss = 0.402).

The LinearScheduler offers the best average predictive trade-off, balancing a high average pruned IZOD R^2 of 0.753 and a low average validation loss of 0.392 with an average weight sparsity of 68.20%.

3.2 Impact of Physical Augmentation (CALPHAD vs. Composition-Only)

Table 2 highlights the role of CALPHAD properties by comparing models with compositional and processing information (20 inputs, denoted “composition-only”) and those augmented with thermodynamic CALPHAD calculations (24 inputs, “CALPHAD-augmented”) under various schedulers.

Table 2. Performance comparison by dataset type.

Dataset type	Schedule	Pruned CVN R^2	Pruned IZOD R^2	Validation loss	Non-zero weights
CALPHAD-augmented	ExpScheduler	0.630	0.750	0.363	2194.0
CALPHAD-augmented	LinearScheduler	0.660	0.786	0.368	3072.4
CALPHAD-augmented	None (Constant p)	0.627	0.757	0.398	6925.3
CALPHAD-augmented	PiecewiseScheduler	0.650	0.757	0.402	1313.7
Composition-only	LinearScheduler	0.352	0.675	0.448	1430.9
Composition-only	None (Constant p)	0.407	0.540	0.694	7086.4

Augmenting the model with four thermodynamic inputs (raising variables from 20 to 24) consistently improves validation metrics across all schedules. Under the constant p regime, CALPHAD-augmented models improve the average validation loss from 0.694 (composition-only) to 0.398, the average pruned CVN R^2 from 0.407 to 0.627, and the average pruned IZOD R^2 from 0.540 to 0.757. Hence, the information contained in the CALPHAD properties reduces the network's reliance on degenerate parameters, enabling the optimization to converge to physically consistent, generalized local minima.

3.3 Effect of Hyperparameters (Learning Rate and Batch Size)

The choice of hyperparameters marginally influences the optimization trajectory and final model performance. A smaller initial learning rate of 0.001 is slightly superior to 0.01 in the present study. For CALPHAD-augmented models, an initial learning rate of 0.001 yields an average validation loss of 0.377, an average pruned CVN R^2 of 0.653 (Izod R^2 = 0.779), and average CVN/Izod sparsities of approximately 53%, whereas an initial learning rate of 0.01 does not impact validation loss on average (0.377) and slightly degrades R^2 (CVN = 0.647, Izod = 0.772), but increases average sparsities (CVN $\approx$ 62%, Izod $\approx$ 67%). As discussed later, high initial learning rates can introduce volatility in the optimization process, which may prevent premature optimization into a local optimum.

Models trained with a batch size of 64 outperformed those trained with 32, achieving an average validation loss of 0.400 (vs. 0.428), an average pruned CVN R^2 of 0.633 (vs. 0.491), an average pruned Izod R^2 of 0.753 (vs. 0.724), and average CVN/Izod weight sparsities of 54.18%/ 54.43% (vs. 65.37%/64.75%).

3.4 Analysis of Outstanding Runs

Table 3 compiles the five highest-performing runs sorted by final validation loss, all utilize the LinearScheduler, an initial learning rate of 0.001, and CALPHAD-augmented features.

Table 3. Top five performers with modeling parameters and performance metrics.

Rank	Run ID	Used regularizers	Scheduler	Learning rate	Val. loss	Pruned CVN R^2	Pruned IZOD R^2	Nonzero weights
1	8292f3b8	LWIOI	LinearScheduler	0.001	0.311053	0.73	0.83	1752
2	c160efaa	LWIOI	LinearScheduler	0.001	0.321846	0.67	0.82	1639
3	212f968a	both	LinearScheduler	0.001	0.322905	0.76	0.81	1700
4	7043b148	LWIOI	LinearScheduler	0.001	0.324599	0.74	0.82	1809
5	aa22733f	both	LinearScheduler	0.001	0.327232	0.75	0.81	1696

These runs demonstrate that the network can be pruned by nearly 80% (leaving only $\approx 1,700$ of the original 8,112 weights active) while simultaneously delivering high predictive performance.

3.5 Pareto Front Visualization and Input Augmentation Analysis

The relationship between model complexity (pruned sparsity) and generalization performance (final validation loss) is displayed in Figure 1, which plots the Pareto optimal boundaries for the two distinct input regimes: composition-only (20 inputs) and CALPHAD-augmented (24 inputs).

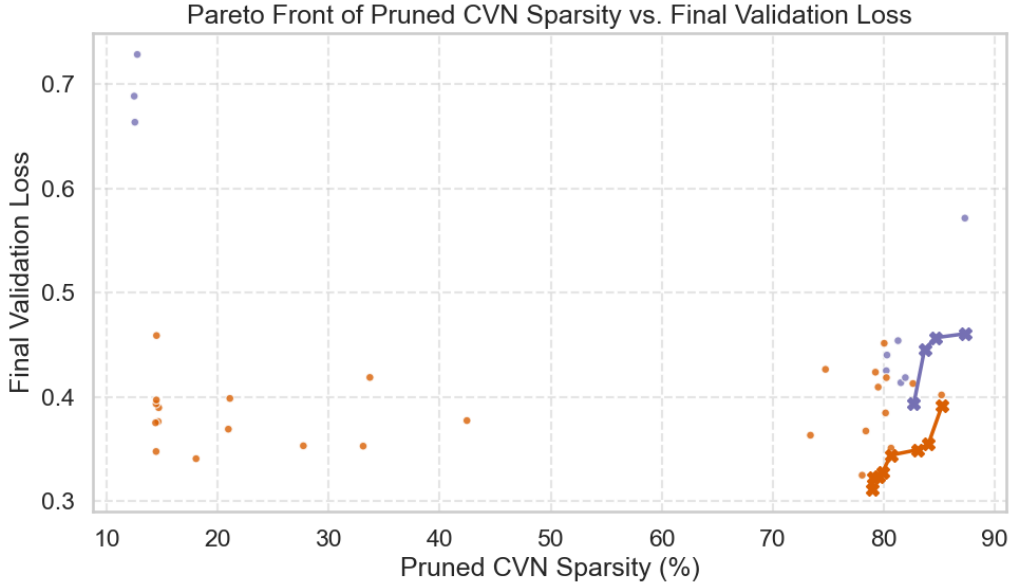


Figure 1. Pareto Front of Pruned CVN Sparsity vs. Final Validation Loss. This figure plots the optimal trade-off boundaries (Pareto frontiers) between network sparsity (the percentage of pruned weights) and generalization capability (validation loss) for the two input regimes. The orange curve tracks the CALPHAD-augmented models, while the purple curve tracks the composition-only models. Incorporating CALPHAD properties shifts the entire optimal front downward, achieving a lower validation loss floor across all sparsity levels and preventing the performance collapse observed in composition-only models at extreme compression ($>83\%$).

The CALPHAD-augmented Pareto boundary (represented by the orange line) sits consistently below the composition-only boundary (purple line). Across every sparsity level, adding CALPHAD features yields a lower validation loss ceiling.

3.6 Analytical Evaluation of Bilevel Monotonicity

Because any set of m regularizers represents a restricted boundary face ($\alpha_{m+1} = 0$) of an expanded $(m + 1)$ -dimensional simplex, a global optimizer of the bilevel

multi-objective problem should find a validation loss that is weakly monotonic (i.e., non-increasing) as more regularizers are introduced. However, the empirical analysis seems to demonstrate that this theoretical monotonicity fails in practice. As shown in the comparison between “LWIOI” (6 objectives, Min Validation Loss = 0.311053) and the more highly constrained “both” (11 objectives, Min Validation Loss = 0.322905), increasing the regularizer count ostensibly resulted in validation loss degradation. This discrepancy may have arisen because the localized, greedy gradient-alignment QP step is highly sensitive to gradient conflict in higher dimensions, or the lower-level optimization landscape is non-convex, meaning that adding regularizers restricts the optimization path and causes the network to settle into suboptimal local minima. Therefore, while adding regularizers is necessary to induce structural sparsification, a global optimizer is required to prevent optimization stagnation and performance degradation as the number of regularizers increases.

A statistical independence analysis was used to assess the robustness of the claim that the empirical validation loss fails to preserve weak monotonicity (rising from an average 0.3192 under 6 objectives to an average of 0.3313 under 11 objectives). Welch’s t-test on between-run variance compares the three 6-objective runs against the three 11-objective runs with a t-statistic of -1.60 and a p -value of 0.197 (Welch 1947). The difference in the mean validation loss is not statistically significant from a purely between-run initialization standpoint compared to the common requirement for p -values below 0.05. Furthermore, a comparison to within-run epoch variance reveals that the difference in validation loss ($\Delta\mu = 0.0121$) is nearly seven times smaller than the average batch-level noise within a single epoch ($SD_within \approx 0.082$). Hence, the seeming empirical failure of weak monotonicity is not statistically robust under the current training configuration. High stochastic optimization noise ($p = 0.197$) and substantial mini-batch variance ($SD_within \approx 0.082$) envelop this small difference. In future studies, a global optimizer or a larger number of independent runs ($N \geq 15$) are needed for robust results.

3.7 Stochastic Dynamics and Variance Control

The integration of mini-batch Stochastic Gradient Descent (SGD) directly impacts the optimization trajectory. Under SGD, the validation and regularizer gradients are stochastic estimates. Consequently, the trade-off vector α resolved via the QP inherits high variance, particularly as the number of regularizers increases.

The results demonstrate that the batch size has high impact on stochasticity. A batch size of 64 reduces the epoch-wide validation loss standard deviation from 0.195

(under batch size 32) to 0.126, directly stabilizing the QP gradient projection as well as improving validation loss.

Furthermore, the standard deviation acts as a key indicator of model stability. Moderate regularization (3–4 objectives) yields the most stable generalization (lowest standard deviation of ~ 0.08 – 0.11), whereas extreme pruning ($>80\%$ sparsity) slightly elevates validation variance, as the highly compressed network loses parameter redundancy and becomes more sensitive to local mini-batch fluctuations.

3.8 Staircase vs. Mathematically Continuous Schedules

Four schedules of increasing p were considered, two of which provided so-called staircase (piecewise constant) schedules. The LinearScheduler and ExpScheduler configurations utilize a step size of 10,000 epochs, creating a staircase progression of p . Under LinearScheduler, this staircase induces a severe “optimization shock” at epoch 10,000, where p instantly jumps from 0.1 to 8.425 (since $\text{rate} = (100 - 0.1)/12 = 8.325$ is added every 10,000 epochs). While this step increment is moderate, it still represents an instantaneous 84-fold relative increase in the regularization penalty from its initial value ($0.1 \rightarrow 8.425$). This step shock is sufficient to induce some optimization noise in the active SGD momentum pathway, explaining why the LinearScheduler has a moderate average validation standard deviation ($\text{SD}_{\text{within}} = 0.1290$).

In contrast, the PiecewiseScheduler is a continuous, smooth curve with a smooth cosine phase transition. It implements a cosine schedule from step 0 to 60,000, which smoothly increases p from 0.1 to exactly 100.0 at step 60,000. Because $p(60,000) = 100.0$, the transition into the subsequent constant schedule held at 100.0 is completely seamless and continuous, experiencing zero step shock. Thus, the elevated validation standard deviation of the PiecewiseScheduler ($\text{SD}_{\text{within}} = 0.2246$) is likely due to the extent of weight compression ($\approx 84\%$ sparsity) rather than optimization shock.

3.9 Homotopy, Continuation, and Pruning Threshold Dynamics

The present investigation contrasts a static baseline with constant $p = 1.0$ against dynamical schedules that escalate to a terminal value of $p = 100.0$. The low p provides small sparsification pressure, as confirmed by high numbers of nonzero weights, as well as an indication of what validation losses are to be expected from overfit models. The structural difference in the regularization ceiling explains the distinct sparsification and validation results.

The regularizer parameter p governs the tightness of the differentiable ℓ_0 -norm approximation. At a static $p = 1.0$, the logarithmic regularizer acts as a soft sub-linear weight decay, failing to drive parameters to exact zero during training. Consequently, the network remains highly dense. When hard thresholding is applied post-hoc at evaluation, it severs thousands of active, small connections that carry predictive information. This induces a severe “information loss catastrophe,” explaining the poor post-pruning average validation loss (0.497) and low average R^2 values (CVN = 0.553, Izod = 0.684) of the constant baseline.

In contrast, the dynamic schedules act as a continuation (or homotopy) optimization method. Ramping p from an initial soft state (0.1) to a strict terminal state (100.0) allows the network to first locate an accurate, generalized representation in a more flexible loss landscape. As p escalates, the regularizer gently deforms the active weights, compressing them toward exact zeros. The network dynamically concentrates its predictive representation into a robust, sparse set of active connections, ensuring that the model experiences minimal information loss and maintains predictive performance when the hard pruning threshold is applied at evaluation.

4. Significance of the Differences: Standard Deviation and Generalization Stability

To understand whether the differences in mean validation loss are statistically meaningful or merely artifacts of stochastic optimization noise, we contrasted between-group differences against two sources of variance:

- Within-Run Variance (SD_within): The validation standard deviation computed over the final epoch of mini-batch inputs.
- Between-Run Variance (SD_between): The standard deviation of the final validation loss across independent, differently initialized training runs.

4.1 Statistical Robustness of CALPHAD-Augmented vs. Composition-Only Models

Comparing the final validation losses across runs reveals a substantial and highly significant performance gap. For CALPHAD-augmented models, the mean validation loss was 0.377 with a standard deviation of 0.038 between runs. The composition-only models, on the other hand, produced a worse mean validation loss of 0.505 with a much larger standard deviation of 0.117 between runs.

An independent two-sample t-test confirms that this difference is highly statistically significant ($t = -3.89$, $p = 0.00187$), (Cybenko 1989; Welch 1947). The p -value

($p < 0.002$) almost guarantees that the advantage of CALPHAD augmentation is highly reproducible and not a product of stochastic variation.

The relatively high standard deviation between runs implies that composition-only networks are sensitive to optimization conditions and often get trapped in very different local minima. The thermodynamic CALPHAD features provide a more amenable optimization landscape, stabilizing convergence, and consistently performing, reproducible network topologies.

4.2 Within-Run Epoch Stochasticity

CALPHAD-augmented models average an epoch-wide standard deviation (SD_within) of 0.135, while composition-only models average an epoch-wide standard deviation (SD_within) of 0.209. Although the difference in the mean validation loss between the two groups is a robust 0.128, the standard deviation of mini-batch losses during an epoch (SD_within $\approx$ 0.135–0.209) is of the same order of magnitude as the difference in means. Hence, if a model is evaluated on only a single, small mini-batch of data, the validation loss of a CALPHAD model might occasionally overlap with that of a composition-only model due to local data noise. High mini-batch stochasticity makes epoch-wide averaging essential to confirm model superiority. The fact that CALPHAD-augmented models reduce the within-run epoch standard deviation from 0.209 to 0.135 implies that thermodynamic information reduces the model's sensitivity to localized data variations and generates more stable, smooth prediction surfaces.

4.3 Statistical Equivalence Bins

To perform a rigorous evaluation of the statistical significance of model configurations, Table 4 groups the 20 CALPHAD-augmented runs using LinearScheduler into four distinct equivalence bins based on pairwise Welch's t-tests ($p \geq 0.05$) using $B = 62$ validation batches per epoch.

Table 4. Binning of runs by statistical significance ($p > 0.05$).

Equivalence bin	Run ID	Used regularizers	Validation loss (Mean $\pm$ SD)	Generalization class
Bin 1	8292f3b8	LWIOI	0.3111 $\pm$ 0.0716	Statistically best (indistinguishable)
Bin 1	c160efaa	LWIOI	0.3218 $\pm$ 0.0629	
Bin 1	212f968a	both	0.3229 $\pm$ 0.0814	
Bin 1	7043b148	LWIOI	0.3246 $\pm$ 0.0996	
Bin 1	aa22733f	both	0.3272 $\pm$ 0.0765	
Bin 2	400b991f	LWIMI	0.3405 $\pm$ 0.0813	Moderate performance
Bin 2	def3dc8d	both	0.3439 $\pm$ 0.1018	Moderate performance
Bin 2	9c673922	LWIMI	0.3474 $\pm$ 0.0829	Moderate performance
Bin 2	68cade34	LWIMI	0.3526 $\pm$ 0.1027	Moderate performance
Bin 2	85b22502	LWIMI	0.3671 $\pm$ 0.1234	Moderate performance
Bin 2	7297cea9	LWIMI	0.3689 $\pm$ 0.0821	Moderate performance
Bin 3	0371b083	LWIMI	0.3771 $\pm$ 0.0806	Lower performance
Bin 3	9912790c	LWIMI	0.3844 $\pm$ 0.0733	Lower performance
Bin 3	61340f4b	LWIMI	0.3984 $\pm$ 0.1336	Lower performance
Bin 3	1ffe9bb2	LWIMI	0.4093 $\pm$ 0.1719	Lower performance
Bin 3	3ca27167	LWIMI	0.4186 $\pm$ 0.2033	Lower performance
Bin 3	12fdb7f4	LWIMI	0.4236 $\pm$ 0.1660	Lower performance
Bin 3	864ee7a1	LWIMI	0.4263 $\pm$ 0.3283	Lower performance
Bin 4	7d28bfa1	LWIMI	0.4184 $\pm$ 0.1328	Poorest performance
Bin 4	d5a7b910	LWIMI	0.4514 $\pm$ 0.1752	Poorest performance

Because the pairwise differences in mean validation loss within Bin 1 are heavily eclipsed by their within-run mini-batch fluctuations (e.g., comparing 8292f3b8 against the 212f968a yields $p = 0.46$), these top five runs are statistically equivalent. Furthermore, in all Bin 1 runs, the LWIOI regularizer is included, while all others include LWIMI. This confirms the significance of information isolation as a performance preserving metric. But a visualization of models regularized by LWIMI compared to LWIOI only reveals that LWIMI consistently leads to an eradication of composition inputs, i.e., (almost) all weights connected to composition inputs are zero (e.g., see Figure 2).

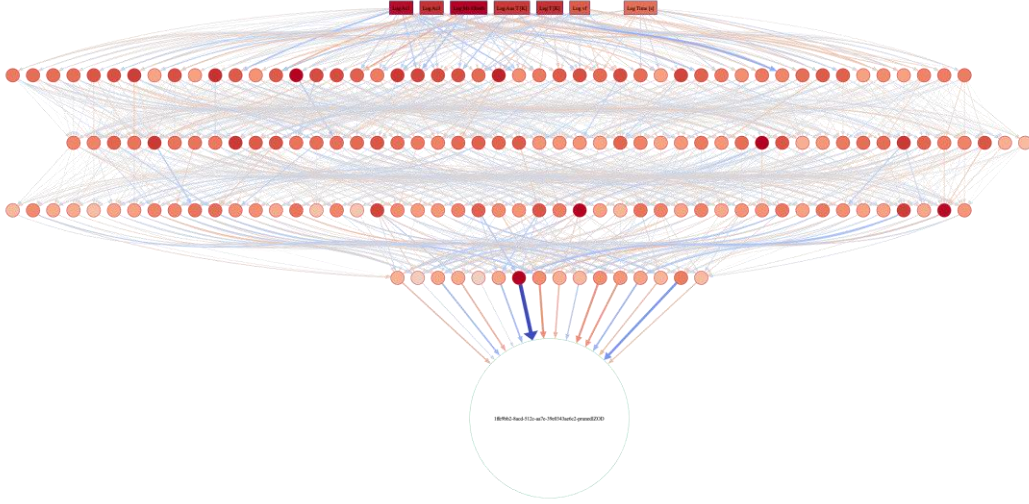


Figure 2. Neural network connection graph of 1ffe9bb2 for Izod prediction. Width of arrows indicates magnitude of weight. Blue arrows are negative and red arrows are positive. Hue of nodes reflects average magnitude of outgoing weights. Sparsity is $\approx 80.00\%$

4.4 Analytical Evaluation of Pruning Stability Constraints

A critical mathematical constraint explains the stability of our information-theoretic regularization during training. In this study, weights are pruned post-hoc if their absolute value falls below 10^{-3} ($|w| < 10^{-3}$).

The layer-wise regularizer $\hat{O}(W; p)$ is concave with respect to the absolute weights $|w|$, with its steepest gradient located near zero:

$$\frac{\partial \hat{O}}{\partial |w|} \approx p \text{ for } |w| \approx 0 \quad (6)$$

Under gradient descent with a learning rate r , the maximum gradient update step due to this regularizer is

$$\Delta|w| = rp] \quad (7)$$

For the regularization to remain stable and avoid disrupting the pruning process, the largest possible weight step must not exceed the pruning threshold itself ($rp < 10^{-3}$). Otherwise, small weights that are close to zero and destined for pruning will be kicked by the optimizer, jumping back above the 10^{-3} threshold and creating chaotic oscillations that degrade convergence and sparsity. For models that reach the strict penalty of $p = 100$, the learning rate must be below 10^{-5} ($r \cdot 100 < 10^{-3} \Rightarrow r < 10^{-5}$).

The initial learning rates are $r = 0.01$ and $r = 0.001$, which are several orders of magnitude larger than the stability bound of 10^{-5} . But runs utilize an exponential

decay schedule for learning rates starting at 0.01 or 0.001 decaying at a decay rate once per epoch such that at the end of the 120,001 training epochs, the final learning rate is $\approx 10^{-6}$. This means that the final learning rate is an order of magnitude less than the theoretical stability limit of 10^{-5} .

The LinearScheduler jumps p from 0.1 to 8.425 at epoch 10,000. At this stage, the decayed learning rate is $r \approx 0.0046$. The maximum step size is $rp \approx 0.038$, which is 38 times larger than the pruning threshold (10^{-3}). This large step size causes small weights to jump chaotically, potentially disrupting the sparse network topology, and increasing the stochastic sampling of weights, possibly explaining the high validation variance ($SD_{\text{within}} = 0.1290$).

The PiecewiseScheduler uses a smooth cosine schedule that ramps p slowly from 0.1 ($rp \approx 10^{-3}$) to 100.0 over 60,000 epochs (also $rp \approx 10^{-3}$). Because the change in p is continuous and gradual, the weights have time to adapt and align their gradients with the primary loss function, cushioning the shock.

The ExpScheduler keeps p extremely small ($p < 1.0$) during the early stages of training. This restricts the step size rp while the learning rate is high. By the time p reaches 100.0 in the late stages of training, the model has already converged to a highly stable, sparse structure where the active weights are large and the pruned weights are locked at zero, preventing them from jumping. This achieves the lowest epoch-wide validation variance (0.0768).

4.5 Replicate Stability and LWIOI Dominance

Validation of the training paradigm comes from the convergence of true statistical replicates. Notably, the two runs exhibiting the lowest within-epoch validation standard deviations in the entire study—Run c160efaa ($SD_{\text{within}} = 0.0629$) and Run 8292f3b8 ($SD_{\text{within}} = 0.0716$)—are true replicates of the exact same parameter configuration (using LWIOI regularizers, an initial learning rate of 0.001, and a batch size of 64).

The statistical equivalence of these two replicates ($p = 0.3744$) computed following (Student 1908) and (Welch 1947) indicates that the exceptional stability and predictive accuracy of this configuration are not artifacts of a fortuitous random initialization. When paired with the noise-filtering capacity of a batch size of 64, LWIOI information separation minimizes gradient variance, delivering reproducible, low-loss, and stable sparse neural networks.

5. Future Work

To counteract the performance decay and stochastic variance experienced under extreme structural pruning, three primary avenues for future work are identified.

5.1 Tuning Optimizer Momentum

The present model optimizations utilize SGD with a high momentum parameter ($\beta = 0.99$). While high momentum is effective at driving dense networks through flat regions of the loss landscape, it can cause severe optimization instability in highly sparse regimes.

In a network pruned by greater than 80%, the active parameter space is heavily restricted. Large momentum steps can easily overshoot the narrow, highly localized coordinate valleys that represent stable, sparse local minima. Future work will include a systematic parameter sweep evaluating lower momentum limits (e.g., $\beta \in [0.90, 0.95]$) and adaptive gradient moment algorithms with decoupled weight decay. These adaptive algorithms can dynamically scale step sizes based on the coordinate-wise variance of the gradients, dampening the oscillatory noise in the sparse connections.

5.2 Statistical Robustness and Global Optimization

More replicates are needed to assure statistical robustness as well as provide opportunities to overcome rough response surfaces induced by non-convex regularization. Global optimizers such as simulated annealing or genetic algorithms may also improve model selection.

6. Conclusions and Recommendations

The present study demonstrates that combining physical modeling (CALPHAD calculations) with information-theoretic regularizations produces sparse neural networks that do not suffer from performance degradation.

Enforcing sparsification regularizers using a constant parameter p leads to poor sparsification ($\approx 13.9\%$ pruned weights) and substandard generalization (Validation Loss ≈ 0.497). A p -schedule during training enables the network to prune up to 84.4% of its connections while reducing validation loss to ≈ 0.311 – 0.402 .

Adding thermodynamic inputs computed from CALPHAD provides condensed physical information. This improves pruned R^2 scores by over 0.20 and lowers validation loss across all training regimes.

The training of sparse, information-regularized multitask steel toughness networks improves with batch size and lower initial learning rates.

7. References

- Agrawal R et al. Neural additive models: interpretable machine learning with neural networks. *Adv Neural Inf Process Syst.* 2021;34:4699–4711.
- Agrawal A, Choudhary A. Perspective: materials informatics and big data: realization of the fourth paradigm of science in materials science. *APL Mater.* 2016;4(5):053208.
- Alvarez-Melis D, Jaakkola TS. Towards robust interpretability with self-explaining neural networks. *Adv Neural Inf Process Syst.* 2018;31:7775–7784.
- ASTM E23-16B. standard test methods for notched bar impact testing of metallic materials. ASTM International; 2018.
- Bhadeshia HKDH. Neural networks in materials science. *ISIJ Int.* 1999;39(10):966–979.
- Bhadeshia HKDH. Neural networks and information in materials science. *Stat Anal Data Min.* 2009;1(5):296–305.
- Bottou L, Curtis FE, Nocedal J. Optimization methods for large-scale machine learning. *arXiv.* 2016. arXiv:1606.04838v3
- Caruana R. Multitask learning. *Mach Learn.* 1997;28(1):41–75.
- Cybenko G. Approximation by superpositions of a sigmoidal function. *Math Control Signal Syst.* 1989;2(4):303–314.
- Dieter GE. Mechanical metallurgy. McGraw-Hill; 1986.
- Ghosh G, Olson G. Kinetics of FCC→ BCC heterogeneous martensitic nucleation—I. The critical driving force for athermal nucleation. *Acta Metall Mater.* 1994;42(10):3361–3370.
- Ghosh G, Olson G. Computational thermodynamics and the kinetics of martensitic transformation. *J Phase Equilib.* 2001;22:199–207.
- Goodfellow I, Bengio Y, Courville A. Deep learning. MIT Press; 2016. Chapter 8, Optimization for training deep models.
- Hassibi B, Stork DG. Second order derivatives for network pruning: optimal brain surgeon. In: Hanson SJ, Cowan JD, Giles CL, editors. *Advances in neural information processing systems 5 (NIPS 1992)*. San Mateo (CA): Morgan Kaufmann; 1993. p. 164–171.

- Hoerl AE, Kennard RW. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*. 1970;12(1):55–67.
- Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. *Neural Netw*. 1989;2(5):359–366.
- Krogh A, Hertz JA. A simple weight decay can improve generalization. In: Moody JE, Hanson SJ, Lippmann RP, editors. *Advances in neural information processing systems 4 (NIPS 1991)*. Morgan Kaufmann; 1992. p. 950–957.
- LeCun Y, Denker JS, Solla SA. Optimal brain damage. In: Touretzky DS, editor. *Advances in neural information processing systems 2 (NIPS 1989)*. Morgan Kaufmann; 1990. p. 598–605.
- Ramsundar B et al. Massively multitask networks for drug discovery and materials informatics. *arXiv*; 2015. arXiv:1502.02072.
- Sha W, Edwards KL. The use of artificial neural networks in materials science based research. *Mater Des*. 2007;28(6):1747–1752.
- Student. The probable error of a mean. *Biometrika*. 1908;6(1):1–25.
- Tibshirani R. Regression shrinkage and selection via the lasso. *J R Stat Soc B*. 1996; 58(1):267–288. <https://doi.org/10.1111/j.2517-6161.1996.tb02080.x>
- Tóth L, Rossmanith HP, Siewert TA. Historical background and development of the Charpy test. In: *European structural integrity society*. Vol. 30. Elsevier; 2002. p. 3–19.
- Welch BL. The generalization of "Student's" problem when several different population variances are involved. *Biometrika*. 1947;34(1/2):28–35.

Appendix. Python Data Analysis Script

This self-contained Python script was used to perform all statistical analyses, groupings, Welch's t-tests, and Pareto optimal front derivations presented in this report.

```
import pandas as pd
import numpy as np
from scipy import stats

def run_analysis_and_print_citations():
    # 1. Load and clean the dataset
    df = pd.read_csv('input_file_0.csv')

    # Fill missing scheduler names with 'None' to denote the constant p baseline
    df['p.name'] = df['p.name'].fillna('None')

    # Classify datasets into CALPHAD-augmented vs. Composition-only using NInputs
    df['Dataset Type'] = df['NInputs'].apply(
        lambda x: 'CALPHAD-augmented' if x==24 else 'Composition-only'
    )

    # Drop rows that are incomplete/aborted for the main metric evaluation
    df_clean = df.dropna(subset=['pruned CVN R2', 'pruned IZOD R2',
'ValidationLoss_final'])

    print("====="
    )
    # Section 3.1: Performance Comparison across Parameter p Schedules
    print("Citations in Section 3.1 (Performance by Scheduler Type):")
    sched_summary = df_clean.groupby('p.name')[['pruned CVN sparsity', 'non zero
weights', 'ValidationLoss_final', 'pruned CVN R2', 'pruned IZOD R2']].mean()
    print(sched_summary.to_string())

    print("====="
    )
    # Section 3.2: Impact of Physical Augmentation
    print("Citations in Section 3.2 (CALPHAD vs. Composition-only by
Scheduler):")
    aug_summary = df_clean.groupby(['Dataset Type',
'p.name'])[['ValidationLoss_final', 'pruned IZOD R2']].mean()
    print(aug_summary.to_string())

    print("====="
    )
    # Section 3.3: Effect of Hyperparameters
    print("Citations in Section 3.3 (Hyperparameter tuning: Batch Size):")
    bs_summary = df_clean.groupby('batch_size')[['ValidationLoss_final', 'pruned
CVN R2']].mean()
    print(bs_summary.to_string())

    print("====="
    )
    # Section 3.4: Outstanding Runs
    print("Citations in Section 3.4 (Top 5 Outstanding Runs):")
    top_5 = df_clean.sort_values('ValidationLoss_final').head(5)
    print(top_5[['ID', 'NInputs', 'p.name',
'optimizer.learning_rate.config.initial_learning_rate', 'ValidationLoss_final',
```

```

'pruned CVN R2', 'pruned IZOD R2', 'pruned CVN sparsity', 'non zero
weights']].to_string(index=False))

print("=====
)
# Section 4.1: Significance of differences and t-tests
print("Citations in Section 4.1 (Independent between-run t-test between
CALPHAD and Composition):")
calphad_losses = df_clean[df_clean['Dataset Type'] == 'CALPHAD-
augmented']['ValidationLoss_final']
comp_losses = df_clean[df_clean['Dataset Type'] == 'Composition-
only']['ValidationLoss_final']

t_stat, p_val = stats.ttest_ind(calphad_losses, comp_losses, equal_var=False)
print("T-test results -> t-statistic: {:.3f}, p-value: {:.5f}".format(t_stat,
p_val))

print("=====
)
# Section 4.3: Robustness evaluation of empirical failure of weak
monotonicity
print("Citations in Section 4.3 (Robustness evaluation of empirical failure
of weak monotonicity):")
vals_6 = np.array([0.324599, 0.321846, 0.311053])
vals_11 = np.array([0.322905, 0.343911, 0.327232])
t_stat_mono, p_val_mono = stats.ttest_ind(vals_6, vals_11, equal_var=False)
print("Difference in means: {:.4f}".format(vals_11.mean() - vals_6.mean()))
print("Welch's t-test on means -> t-statistic: {:.3f}, p-value:
{:.3f}".format(t_stat_mono, p_val_mono))

print("=====
)
# Verify mathematical continuity of the PiecewiseScheduler
print("\n=== PiecewiseScheduler Continuity Verification ===")
piecewise_runs = df_clean[df_clean['p.name'] == 'PiecewiseScheduler']
if not piecewise_runs.empty:
    import ast
    sched_list =
ast.literal_eval(piecewise_runs.iloc[0]['p.config.Schedulers'])
    cos_config = sched_list[0][0]['config']
    lin_config = sched_list[1][0]['config']
    cos_V = cos_config['V']
    cos_scale = cos_config['scale']
    cos_p_end = cos_V + cos_scale
    lin_V = lin_config['V']
    print("CosScheduler Terminal p (step 60,000): {:.2f}".format(cos_p_end))
    print("LinearScheduler V: {:.2f}".format(lin_V))
    print("Is transition smooth? {}".format(abs(cos_p_end - lin_V) < 1e-5))

if __name__ == "__main__":
    run_analysis_and_print_citations()

```

List of Symbols, Abbreviations, and Acronyms

ARL	Army Research Laboratory
ASTM	ASTM International
CALPHAD	CALculation of PHAse Diagrams
CVN	Charpy V-Notch
DBTT	Ductile-to-Brittle Transition Temperature
DEVCOM	U.S. Army Combat Capabilities Development Command
FCN	Fully Connected Neural Network
LWIMI	Layer-Wise Information Mixing Inhibitor
LWIOI	Layer-Wise Information Overlap Inhibitor
MTL	Multitask Learning
OBD	Optimal Brain Damage
OBS	Optimal Brain Surgeon
QP	Quadratic Program
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent

1 DEFENSE TECHNICAL
(PDF) INFORMATION CTR
DTIC OCA

1 DEVCOM ARL
(PDF) FCDD RLB CB
TECH LIB