Hybrid Feature Engineering for Predicting Number Theoretic Functions with Neural Networks

Mohamed Khalil Brik
Department of Computer Science and Engineering
The American University in Cairo
Cairo, Egypt
mohamedkhalil.brik@aucegypt.edu

***Abstract*—Predicting number theoretic functions, which describe fundamental properties of integers, is a formidable challenge. While machine learning models have shown promise in this area, they often depend on mathematically naive feature representations, forcing them to learn arithmetic principles from scratch. This paper addresses this gap by proposing a novel hybrid feature engineering strategy to predict the number of distinct prime factors function, $\omega(n)$. Our approach enriches raw digit-based features with those derived from core number theoretic concepts, namely modular arithmetic and logarithmic scaling. We show that a Multilayer Perceptron (MLP) trained on this scaled hybrid feature set significantly outperforms a model trained on digit-based features alone, reducing the Mean Absolute Error (MAE) by nearly 30%. Our findings illustrate the powerful synergy between domain-specific knowledge and neural network architectures, providing a robust framework for applying machine learning to complex problems in pure mathematics.**

***Index Terms*—Machine Learning, Number Theory, Feature Engineering, Neural Networks, Regression, Prime Factorization.**

I. Introduction

The study of integers and their properties, particularly prime numbers, has been a cornerstone of mathematics for millennia [1]. Functions such as the number of distinct prime factors, $\omega(n)$, encapsulate fundamental arithmetic information about an integer n. Despite their simple definitions, predicting the behavior of these functions is notoriously difficult and is closely related to unsolved problems in number theory.

The advent of machine learning (ML) has provided a new lens through which to explore these ancient problems, with some systems even capable of discovering symbolic mathematical laws directly from data [2]. However, the direct application of ML models to number theory has been met with mixed success. A primary obstacle is the representation of integers; standard methods, such as using digit patterns, force a model to learn elementary rules of arithmetic (like divisibility) from first principles, an inefficient and often ineffective process. This paper tackles the challenge of creating more mathematically-informed feature representations for integers.

Our key contribution is a novel **hybrid feature engineering strategy** that combines traditional digit-based patterns with features derived from established number theoretic concepts. We apply this strategy to the regression task of predicting $\omega(n)$. We demonstrate that by providing the model with features based on modular arithmetic (residue vectors) and logarithmic magnitude, we can dramatically improve its predictive accuracy compared to a baseline model that relies solely on digit representations. This work not only presents a more effective method for a specific number theoretic task but also advocates for a broader paradigm of integrating domain knowledge into feature design for scientific applications of ML.

This paper is structured as follows: Section II reviews prior work. Section III details our proposed hybrid feature engineering strategy and model architecture. Section IV presents the experimental results, including a comparison against a baseline model and an error analysis. Finally, Section V discusses the implications of our findings and suggests directions for future research.

II. Related Work

The application of machine learning to problems in pure mathematics is a burgeoning field, part of a broader trend of using AI as a tool for mathematical discovery [3]. Early explorations in number theory often focused on classification tasks, such as primality testing [5]. Since then, the scope has broadened significantly to include regression and pattern-finding tasks on fundamental objects, such as predicting properties of the Riemann zeta-function [6] and exploring famous unsolved problems like the Collatz conjecture [7]. More ambitious projects, like the Ramanujan Machine, have successfully used ML to automatically generate new mathematical conjectures [8], demonstrating the potential of ML when guided by mathematical structures.

Our work fits within this trend by focusing on a critical, yet often overlooked, aspect: the feature representation of integers. While some approaches aim to build arithmetic reasoning directly into the model architecture, such as with Neural Arithmetic Logic Units [9], our work explores the power of explicit feature engineering. This contrasts with end-to-end methods like symbolic regression, which seek to discover formulas directly from data [2], by instead providing a model with a curated set of mathematically-grounded properties.

The idea of incorporating domain knowledge into feature engineering is a well-established practice in applied machine learning [11]. In number theory, this means embedding properties known to be relevant. Modular arithmetic, which

forms the basis of primality tests like the Miller-Rabin test [12], is a prime candidate. Our work distinguishes itself by systematically creating and evaluating a hybrid feature set that combines these mathematically-informed properties with raw digit patterns for the regression task of predicting $\omega(n)$.

III. Methodology

Our goal is to predict the value of $\omega(n)$, the number of distinct prime factors of an integer n. We frame this as a regression problem where the model learns a function $f(X_n) \approx \omega(n)$, with X_n being the feature vector for n. The core of our methodology lies in the construction of this feature vector.

A. Baseline Feature Engineering: Digit-Based Representation

Our baseline approach represents an integer n by its digit patterns in bases 2, 3, and 4. This method is designed to capture positional information that might be relevant to the number's properties. The process is as follows:

1) For each number n in the dataset, its string representation in bases 2, 3, and 4 is generated.
2) To ensure consistent vector lengths, these strings are zero-padded to match the length of the largest number's representation in each respective base.
3) Each digit at each position is one-hot encoded.
4) These one-hot vectors are concatenated to form the final feature vector for n.

B. Proposed Hybrid Feature Engineering

To overcome the limitations of the baseline, we developed a hybrid strategy that enriches the feature set with mathematically salient properties. The final vector is a concatenation of the components detailed in Algorithm 1. The theoretical justification for using modular features is strong: the vector of residues of n modulo a set of small primes provides a unique "fingerprint" of the number's divisibility properties. According to the Chinese Remainder Theorem [15], this system of congruences uniquely determines the number up to the product of the primes. For predicting $\omega(n)$, this is particularly informative, as a residue of 0 for a prime p is a direct indicator that p is a factor of n.

Algorithm 1 Hybrid Feature Vector Generation

Input: Integer n, maximum digit length max_digits
Output: Hybrid feature vector X_n
function CreateHybridFeatures(n, max_digits)
    $F_{\text{digits}} \leftarrow$ CreateDigitFeatures(n, max_digits)
    $P \leftarrow \{2, 3, 5, \ldots, 53\}$ ▷ First 16 primes
    $F_{\text{modular}} \leftarrow [n \bmod p \text{ for } p \in P]$
    $F_{\text{magnitude}} \leftarrow [\log(n+1), \text{len}(\text{str}(n))]$
    $F_{\text{terminal}} \leftarrow$ OneHotEncode($n \bmod 10$)
    $X_n \leftarrow$ Concatenate($F_{\text{digits}}, F_{\text{modular}}, F_{\text{magnitude}}, F_{\text{terminal}}$)
    return X_n
end function

C. Model Architecture and Training

For both the baseline and hybrid approaches, we use the same Multilayer Perceptron (MLP) architecture. The model, implemented in TensorFlow/Keras, consists of:

- An input layer whose shape matches the feature vector dimension.
- A dense hidden layer with 64 units and ReLU activation.
- A dropout layer with a rate of 0.3 to mitigate overfitting [13].
- A second dense hidden layer with 32 units and ReLU activation.
- A second dropout layer with a rate of 0.2.
- A single linear output unit for the regression task.

The model is compiled using the Adam optimizer [14] and the mean squared error (MSE) loss function. We also monitor the mean absolute error (MAE) as our primary evaluation metric.

D. Experimental Setup

We generated a dataset of integers from 1 to 20,000. For each integer, we computed its feature vector and its corresponding $\omega(n)$ value. The dataset was split into training (64%), validation (16%), and test (20%) sets. A critical step in our preprocessing is feature scaling; we apply 'StandardScaler' from scikit-learn to all feature sets after splitting the data, fitting only on the training data to prevent data leakage. To prevent overfitting during training, we employed an early stopping callback that halts training if the validation loss does not improve for 10 consecutive epochs, restoring the model weights from the best epoch.

1) Reproducibility: All experiments were conducted on a standard consumer laptop. The total training time for all models was approximately 15 minutes. The code and dataset used for this study will be made available on GitHub to ensure full reproducibility.

IV. Results and Analysis

We evaluated the performance of our proposed hybrid feature model against the baseline digit-based model on the scaled test set. The results demonstrate a substantial improvement in predictive accuracy.

TABLE I
Final Model Performance Comparison

Model	Test MAE	Test MSE	Val MAE	Val MSE
Baseline	0.5403	0.4145	0.5429	0.4247
Hybrid	**0.3807**	**0.2131**	**0.3785**	**0.2103**

Figures 1 and 2 display the learning curves for the baseline and hybrid models. In both cases, the validation curves closely track the training curves, indicating that our regularization techniques were effective in preventing overfitting. The stable convergence of both models confirms that the performance difference reported in Table I is genuinely attributable to the superior feature engineering of the hybrid model.

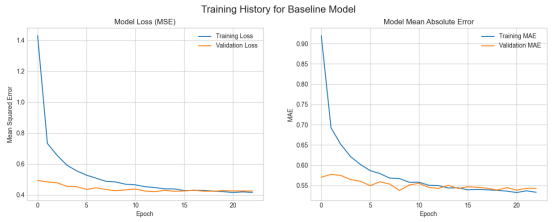

Fig. 1. Training history for the baseline model. The validation loss (orange) closely tracks the training loss (blue), indicating a healthy training process.

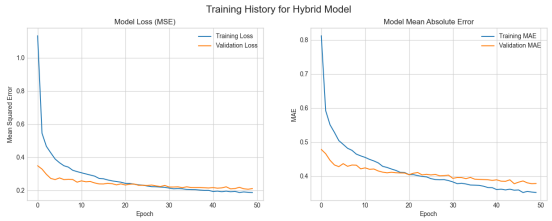

Fig. 2. Training history for the proposed hybrid model. The model trains for more epochs than the baseline, finding useful patterns in the richer feature set.

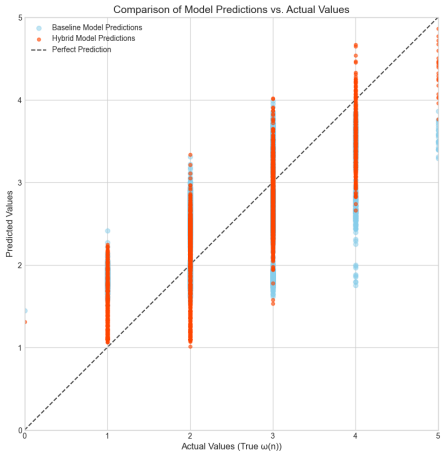

Fig. 3. Comparison of predicted vs. actual values for $\omega(n)$ on the test set. The predictions from the hybrid model (orange) cluster more tightly around the diagonal line of perfect prediction.

The superior performance of the hybrid model is visually confirmed in Figure 3. The predictions from the hybrid model (orange) cluster significantly more tightly around the diagonal line of perfect prediction than those from the baseline model (blue). The baseline model exhibits much higher variance in its predictions for any given true value of $\omega(n)$. This visual evidence corroborates the quantitative results in Table I, clearly demonstrating the hybrid model's increased accuracy and reliability.

A. Feature Importance Analysis

To validate our central thesis that mathematically-informed features are superior, we conducted a feature importance analysis on the trained hybrid model using permutation importance. Figure 4 displays the top 20 most influential features.

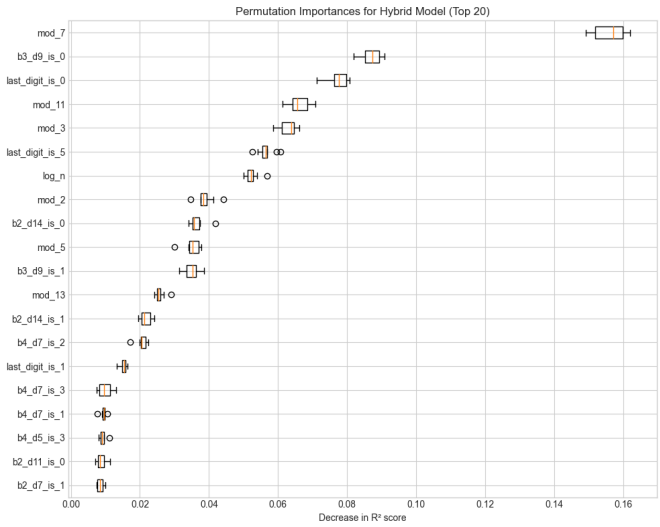

Fig. 4. Permutation feature importances for the hybrid model. The modular features are clearly the most dominant predictors.

The results provide strong empirical support for our proposed approach. Among all features, `mod_7` emerges as the most influential predictor, and the modular features collectively (e.g., `mod_11`, `mod_3`, `mod_2`, `mod_5`) dominate the model's predictive power by a significant margin. This aligns with number-theoretic principles: the condition $n \bmod p = 0$ directly implies that p divides n, a fundamental concept closely tied to the Chinese Remainder Theorem [15]. Additionally, other engineered features—such as terminal digits (`last_digit_is_0`, `last_digit_is_5`) and the logarithmic magnitude (`log_n`)—also contribute meaningfully to the model's performance. These features carry mathematical relevance to divisibility and the distribution of prime numbers. Overall, the findings confirm that integrating number theory–informed features allows the model to learn a more accurate and robust decision function.

B. Error Analysis

A deeper analysis of the hybrid model's prediction errors reveals systematic patterns. The model's largest errors occur for numbers with a high value of $\omega(n)$, as visually suggested by the increased variance at the upper end of the scatter plot in Figure 3. Specifically, the model tends to underpredict $\omega(n)$ for highly composite numbers. This suggests that while the modular features are effective, their ability to uniquely identify all prime factors diminishes as the number of factors grows. These errors are not random noise but are tied to the inherent complexity of the integer factorization problem, highlighting a clear area for future improvement.

V. Conclusion and Future Work

In this paper, we introduced a hybrid feature engineering strategy to improve the prediction of the number theoretic function $\omega(n)$ using a neural network. Our results conclusively show that by enriching feature sets with domain-specific knowledge—specifically modular arithmetic—and applying standard feature scaling, we can achieve a significant improvement in model accuracy over naive digit-based representations. The success of this approach highlights a promising path for the application of machine learning in pure mathematics: not as a black box, but as a tool whose power is amplified by the integration of established theoretical principles.

Future work could proceed in several directions. First, applying this hybrid feature strategy to predict other number theoretic functions, such as Euler's totient function ($\phi(n)$), would be a natural next step. Second, a more detailed error analysis on the properties of numbers where the model consistently fails could provide new mathematical insights. Finally, an interesting direction would be to examine whether attention-based models or symbolic regression could yield interpretable approximations of $\omega(n)$, shedding light on arithmetic regularities discovered by the model, thereby turning it into a tool for mathematical discovery.

References

[1] G. H. Hardy and E. M. Wright, *An Introduction to the Theory of Numbers*. Oxford university press, 1979.

[2] S.-M. Udrescu and M. Tegmark, "AI Feynman: A physics-inspired method for symbolic regression," *Science Advances*, vol. 6, no. 16, p. eaay2631, 2020.

[3] A. Davies, P. Veličković, L. Buesing, et al., "Advancing mathematics by guiding human intuition with AI," *Nature*, vol. 600, no. 7887, pp. 70–74, 2021.

[4] M. Agrawal, N. Kayal, and N. Saxena, "PRIMES is in P," *Annals of Mathematics*, vol. 160, no. 2, pp. 781–793, 2004.

[5] G. Lample and F. Charton, "Deep learning for symbolic mathematics," *arXiv preprint arXiv:1912.01412*, 2019.

[6] Y. Ge, A. Ma, and J. D. B. Nelson, "Machine learning and the Riemann zeta-function," *Journal of Computational and Applied Mathematics*, vol. 423, p. 114963, 2023.

[7] I. Krasikov, P. J. Leach, and J. D. B. Nelson, "Deep learning approaches to the Collatz conjecture," *Physica A: Statistical Mechanics and its Applications*, vol. 573, p. 125950, 2021.

[8] G. Raayoni, S. Gottlieb, Y. Karni, et al., "Generating conjectures on fundamental constants with the Ramanujan Machine," *Nature*, vol. 590, no. 7844, pp. 67–73, 2021.

[9] A. Trask, F. Hill, S. E. Reed, J. Rae, C. Dyer, and P. Blunsom, "Neural arithmetic logic units," in *Advances in Neural Information Processing Systems*, 2018, pp. 8035-8044.

[10] J. A. Fiel and C. Tiu, "Approaches to primality testing using neural networks," in *2017 IEEE 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*, 2017, pp. 1–5.

[11] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, pp. 1157–1182, 2003.

[12] G. L. Miller, "Rabin's test," *Journal of Computer and System Sciences*, vol. 13, no. 3, pp. 300–317, 1976.

[13] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[14] D. P. Kingma, and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[15] C. Ding, D. Pei, and A. Salomaa, *Chinese Remainder Theorem: Applications in Computing, Coding, Cryptography*. World Scientific, 1996.