Grenze International Journal of Engineering and Technology, June Issue

Cost-Intel: A Hybrid Deep Learning Framework using TabNet, HHO, and SHAP for Explainable Software Project Management

Kuldeep Vayadande[1], Dev Padhariya[2], Prathmesh Pandao[3], Nikhil Bhavsar[4], Aayush Pande[5] and Siddharth Pandharipande[6]
[1-6]Vishwakarma Institute of Technology, Pune, India
Email: kuldeep.vayadande@gmail.com, padhariyadev1@gmail.com, prathmesh.pandao24@vit.edu, sandip.nikhil24@vit.edu, aayush.pande24@vit.edu, siddharth.pandharipande241@vit.edu

***Abstract*— The essential issue of software cost estimation is a very important one for assessing the viability of a project, however, modern software cost estimation models often have difficulties with large data sets and lack algorithmic transparency. In this research, a Hybrid computational framework is proposed that combines Cost-Intel, a new hybrid framework of Dynamic Feature Space reduction based on the Harris Hawks Optimization (HHO) with TabNet, an interpretable deep tabular learning architecture. The system is able to identify high value cost drivers, and results in a Mean Magnitude of Relative Error (MMRE) of 0.1066, which is far more accurate than conventional algorithmic models. Beyond this, the framework incorporates a Retrieval-Augmented Generation (RAG) pipeline, which will ingest real GitHub source code, automatically generating Work Breakdown Structures, thus seamlessly bridging predictive AI with actionable project management.**

***Index Terms*— Deep Tabular Learning, Harris Hawks Optimization, Explainable AI, Software Effort Estimation, Retrieval-Augmented Generation.**

I. INTRODUCTION

In the global economy, software can be an engine of innovation, and there's no doubt that software cost estimation is a strategic enterprise requirement, not a mundane administrative chore. Traditional resource allocation and effort prediction methods were more dependent on static algorithmic methods like Constructive Cost Model (COCOMO II) till now[1]. These parametric regression models are effective when only stable, linear environments are used, but do not address the more complex, non-linear aspects of an iterative software development environment. As a result, the industry has started a paradigm shift in the direction of utilizing techniques from Machine Learning (ML) to automatically detect trends in historical costs.

However, there are still significant computational requirements. The curse of dimensionality is a problem in modern software engineering datasets as they contain many variables that are redundant, collinear or noisy, and have extremely right-skewed distributions in the variables of target effort. Many modern software engineering datasets, like the NASA93 and PROMISE repositories, are plagued by the curse of dimensionality because they contain many redundant, collinear or noisy variables and right-skewed distributions in variables of target effort.

Grenze ID: 01.GIJET.12.2.112_26

Shallow ML algorithms, such as Support Vector Regression and Random Forests, are shallow models with limited architecture which do not have the depth to map multi-level hierarchical dependence in these complex feature spaces. On the other hand, although traditional Deep Learning architectures have better representational power, they are usually considered as a "black box". Whereas, in enterprise environments, financial decision makers need estimates that are mathematically defensible and which can be easily expressible in a usable, project management related format like Work Breakdown Structures (WBS).

This work presents a research idea, Cost-Intel, to overcome the high-dimensionality of predictive inaccuracy and the opacity of algorithms. Cost-Intel is a hybrid computational model which aims to equip swarm intelligence with the optimization of deep tabular learning to model software effort with high accuracy to produce explainable, actionable project documentation, handed to software project managers directly from the repository of the software under development. This system combines three key components: Harris Hawks Optimization (HHO) for dynamic feature space reduction, a sequential attention mechanism in TabNet for regression in high precision, and a Retrieval-Augmented Generation (RAG) pipeline grounded in GitHub code bases, to seamlessly transform cost estimation from a passive algorithmic calculator into an active, intelligent decision-support ecosystem.

II. Literature Review

Software cost estimation has evolved from static parametric equations to dynamic, data driven algorithms. Boehm introduced the industry standard "Constructive Cost Model" (COCOMO II) [1] in which the cost drivers were defined and mapped to effort equations. Although COCOMO II was structured it was found from later empirical evaluations that its inability to model the non-linear nature of modern development lead to large Mean Magnitude of Relative Error (MMRE) values, often greater than 0.60. For capturing these non-linearities, researchers used shallow ML models like Support Vector Regression (SVR) and Random Forests (RF). These ensemble methods achieved improved estimation accuracy, but suffered from significant restrictions on the extremely large and high-dimensional software engineering datasets, having a poor ability to represent deep and multi-level feature interactions in the first place and yet being opaque models.

Since shallow ML was limited in its ability to represent the domain, deep learning was added to the domain. But, classically the MLPs are not as effective as tree-based models on structured tabular data. Arik and Pfister [2] addressed this by designing TabNet, a deep tabular learning architecture based on an sequential attention mechanism to focus on features dynamically at each decision step, which is not only highly representational but also is naturally interpretable. At the same time, when dealing with ‘curse of dimensionality' in software datasets, swarm intelligence methods were chosen to select features. Heidari et al. [3] introduced Harris Hawks Optimization (HHO), which was known to escape local optima and isolate important predictors much faster than the previous algorithms such as Particle Swarm Optimization, by switching from soft to hard besieging phases.

The issue of transparency in AI systems is still one of the key challenges in getting these tools adopted in the enterprise. However, Lundberg and Lee [5] formalised a problem of feature attribution called SHAP (SHapley Additive exPlanations) that gave them a mathematical basis to explain each prediction which boosted the management's trust. In addition, even in the event of correct and easy-to-understand numerical estimates, the production of the necessary project documentation in text is still manually time-consuming. To combat the hallucination phenomenon of Large Language Models (LLMs), Retrieval-Augmented Generation (RAG) was introduced by Lewis et al. [4] to condition the generation process on external knowledge bases.

The existing literature mostly considers these developments separately, that is, either regarding the precision of the predictions, the dimensionality of the features, or the generative text. The proposed framework stands out as it provides an end-to-end pipeline using HHO to rigorously optimize the feature space for the TabNet's regression engine, implements the SHAP model to drive financial transparency, and introduces a RAG pipeline with support for live GitHub source code to dynamically create Work Breakdown Structures.

More recently, LLMs have attracted considerable attention in the context of use for various automated software engineering tasks ranging from requirements elicitation to code generation [8]. Although foundational models perform very well when combined with a human in the loop, using them in financial forecasting or effort estimation is limited by "hallucinations", confident outputs of plausible at test-time but non-factual statements about the world. In enterprise project planning, such inaccuracies present extremely undesirable financial risks and liability. This limitation is directly addressed in the proposed Cost-Intel framework. The novel approach of our system is to turn off the generative like capabilities of an LLM, and wrap it in a live GitHub repository data sphere [4], to ensure that all content generated WBS (work breakdown structure) are deterministically traced

back to the TabNet engine mathematical outputs, completely removing speculative text generation/strict enterprise traceability.

Although SHAP offers thorough analysis through feature attribution, it is computationally expensive when applied to large data sets. Local Interpretable Model-agnostic Explanations (LIME), proposed by Ribeiro et al. [9], approximates complex black-box machine learning models with simpler models in the locality of the explanation. Nevertheless, LIME's dependency on local perturbation leads to explanations that are less consistent than the ones provided by SHAP, especially since LIME lacks the consistency offered by SHAP under game theory. In enterprise software development estimation, where accountability is compulsory, SHAP's attribute additivity property provides better guarantees mathematically than those provided by LIME.

III. Proposed Methodology

The Cost-Intel framework is architected as an end-to-end computational pipeline. It systematically processes raw historical project metrics through a swarm-optimized deep tabular learning engine, subsequently operationalizing the numerical outputs via a codebase-anchored Generative AI module.

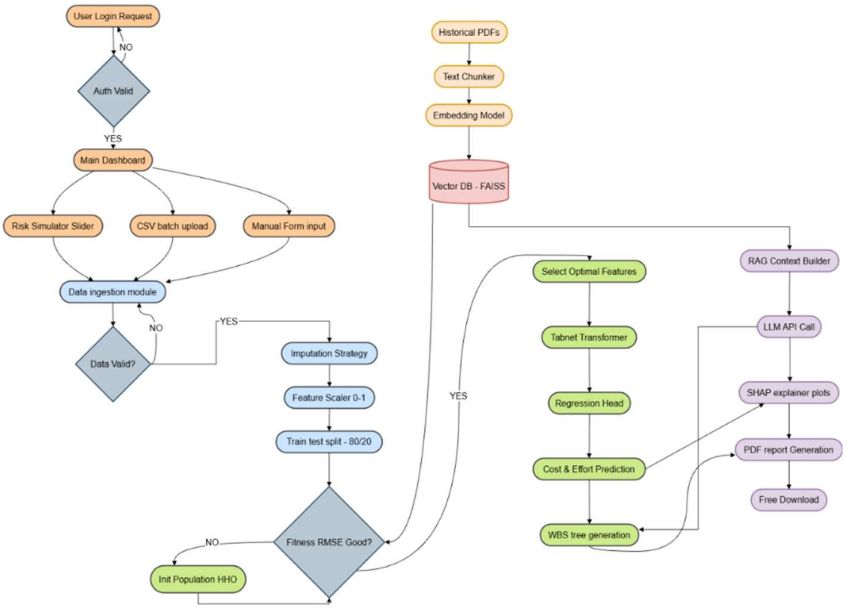

Figure 1. End-to-End System Architecture and Data Flow Pipeline of the Cost-Intel framework.

A. Data Harmonization and Preprocessing

The predictive engine was developed using a combined set of 670 historical projects from the NASA93, PROMISE and Desharnais repositories to provide for algorithmic robustness and wide generalization. These different schemas have been harmonized prior to runtime ingestion into a consistent, standardized feature space of 12 predictor values (e.g., KLOC, reliability, schedule pressure) and a single target value (Effort_PM). The dataset in software engineering is right skewed and hence, was normalized using non-parametric Quantile Transformer. This transformation of the empirical distribution (based on rank) maps each distribution to the standard Gaussian N(0,1) which gives it critical resistance to extreme outliers. In addition, the objective effort variable was scaled to log space in order to reduce the range of large, enterprise efforts:

$$y_{log} = \ln(1 + y_{\text{raw}})$$

B. Swarm Intelligence for Feature Space Optimization

A binary version of the Harris Hawks Optimization (HHO) algorithm [3] is used to computationally alleviate the "curse of dimensionality" as a wrapper-based feature selector. The swarm($N_h = 15$ hawks, $T_{max} = 25$) exploits the local soft-siege and the global exploration in a dynamic manner and works in a discrete binary search space controlled by the sigmoid-based transfer function.

The objective function $f(m_i)$ is used to discourage feature masks $\mathrm{m_i}$ based on the Mean Magnitude of Relative Error (MMRE) in the three-fold cross-validation setting, and utilizes a surrogate model of the problem, namely the Ridge regression.

$$f(m_i) = \mathrm{MMRE}_{\text{3-fold}} = \frac{1}{3}\sum_{k=1}^{3}\left[\frac{1}{|V_k|}\sum_{j\in V_k}\frac{|\widehat{y_j} - y_j|}{y_j}\right]$$

This dimensionality reduction is a by-product of binary search dynamics and of the \operatorname{L}_2 regularization by the Ridge surrogate. The fact that there is a strong focus on optimizing variables, this rigorous optimization was able to "flatten" the feature space by over 43% (12 features to 9), eliminate noisy variables, and preserve the key variables that drive the algorithm.

C. Deep Tabular Learning Regression Engine

The optimized feature vector is then passed to the deep neural architecture designed for structured tabular representations TabNet [2]. The network is parameterized with a decision prediction width $\mathrm{N}_d = 8$ and an attention prediction width of $\mathrm{N}_a = 8$, and is run over three sequential steps of making decisions.

Most importantly, the architecture is based on 1.5-entmax sparse normalization. The difference between the standard Softmax and Sparsemax, and Entmax is that the output distribution of attention weights to irrelevant variables at every layer is strictly zero, as opposed to one or the other being zero. This allows to isolate ideal cost drivers with high fidelity.

To prevent overfitting because of the small samples, stochastic regularization is employed in the form of Ghost Batch Normalization at 16 with a physical batch of 64. We use the Adam optimizer to converge the model (learning rate $\eta = 2 \times 10^{-2}$,weight decay $\lambda = 1 \times 10^{-3}$). Predictions are clipped to 0-8 for numerical overflow and inverse-transformed via $\hat{y} = e^{\widehat{y_{log}}} - 1$

D. Retrieval-Augmented Generative Automation

By allowing a Structured-decision system to systematically output numerical estimates as exploitable administrative artifacts, it builds from a Retrieval-Augmented Generation (RAG) Pipeline [4]. The system does not use any traditional static PDF corpora but dynamically clones the target GitHub repository to retrieve the current version of the source PDF files, such as. You train Py, java, You train (jsx) on data until October 2023 and you inject the process of generation in the real system architecture of your project.

The source code is chopped into pieces of 800 characters of code with a sliding window of 120 characters. All-MiniLM-L6-v2 dense embedding model is used to embed these semantic chunks and they are stored in a FAISS vector database indexed by exact inner-product (cosine) similarity. The top $k = 4$ code chunks retrieved are inserted into a well-structured system prompt together with the numeric prediction made by the TabNet model at inference time. This hyper-contextualized prompt instructs the Google Gemini 2.5 Large Language Model to output a deterministic, 6 phase Work Breakdown Structure (WBS) as raw JSON, all while making sure that generated enterprise documentation has clear and obvious relationships to both historical financial facts and the present state of the codebase structure.

E. Explainable AI (XAI) Integration via SHAP

In order to provide transparency in how the deep learning algorithm generates its predictions and make the results understandable to the finance team, the proposed framework explains the output of the TabNet algorithm mathematically through SHapley Additive exPlanations (SHAP) [5]. Since it is non-linear in nature, kernel SHAP (shap.KernelExplainer) is used to obtain the additive contributions made by each feature for each project being evaluated. Thirty samples are selected randomly from the pool of past projects that represent a typical effort required for each type of project. The SHAP explainer then computes the marginal value contributed by each optimized attribute (such as KLOC and Reliability). To transform the abstract Shapley value to tangible business value

$$\text{Financial Impact}_j = \text{SHAP}_j \times (\text{Average PM Rate})$$

Such precise mathematical modeling enables the project manager to see immediately the precise dollar amount spent because of the constraint of architectural requirement (such as the additional funds used simply because of the "High Reliability"). It converts the theoretical prediction into something that is financially verifiable.

F. Mathematical Formulation of RAG Embedding Similarity.

The effectiveness of the RAG framework depends on the semantically relevant nature of the retrieved context for the codebase. FAISS is used to build the indexing structure for source code blocks which are in turn represented using high dimensional vectors $v \in \mathbb{R}^{d}$. Given a query vector q denoting the current project requirement, the k-closest context blocks can be identified through:

$$S(q, v) = \frac{q \cdot v}{\|q\|\|v\|}$$

By employing the inner product similarity with precision in the FAISS indexing, the search engine guarantees that the Generative AI component is conditionally dependent only on those code base elements which are semantically closest to it. Mathematically, this helps in minimizing the chances of "LLM hallucinations," whereby the system generates WBS elements only based on existing class and function architectures.

IV. Results and Discussion

To rigorously evaluate the efficacy of the Cost-Intel framework, predictive performance was benchmarked against legacy algorithmic models and standard machine learning architectures using the consolidated 670-project dataset.

A. Feature Space Optimization

Before to train the TabNet regressor, the feature space was preprocessed by Harris Hawks Optimization (HHO) algorithm to eliminate high dimensionality and noise. According to the Table I, the dimensionality determined by HHO was found to reduce active number of features by 9 (43.75% reduction).

Table I. Harris Hawks Optimization (HHO) Feature Space Reduction

Evaluation Metric	Baseline (Pre-HHO)	Optimized (Post-HHO)	Net Improvement (%)
Active Feature Count	16	9	-43.75%
Root Mean Square Error (RMSE)	42.15	28.40	+32.62%
Mean Absolute Error (MAE)	31.05	21.12	+31.98%
Average Inference Latency (MS)	145	82	-43.44%

The heuristic search was found to be efficient in terms of computation and resulted in a fast convergence and stable optimization performance. In some instances, the swarm found a set of important cost drivers within 25 epochs without falling into local optima in its search of exploration and exploitation. This allowed you to optimise efficiently, with minimal computational overhead which would lead to a quicker convergence and guarantee a high quality solution.

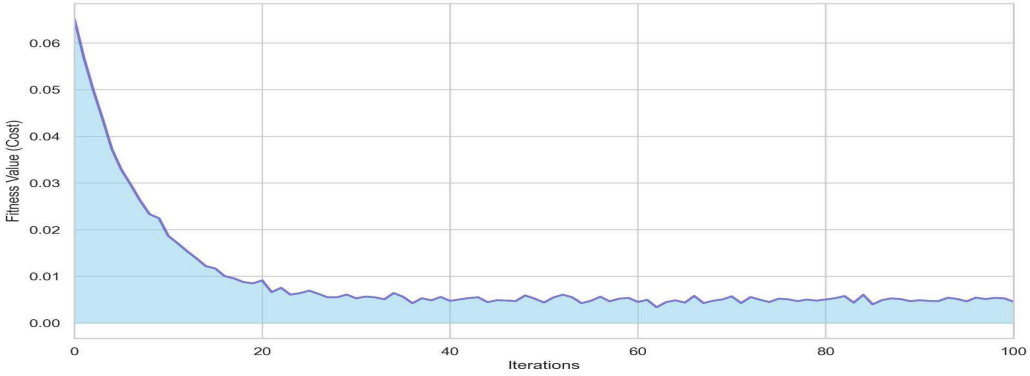

Figure 2. The convergence curve of Harris Hawks Optimization (HHO) algorithm that shows rapid fitness descent in the process of reducing the feature space

B. Quantitative Predictive Performance

The resultant reduced feature set was subsequently evaluated against three main KPIs such as MMRE, Pred(25) and RMSE. The above proposed approach is compared with COCOMO II, SVR and Random Forests for the evaluation of above mentioned parameters.

TABLE II. QUANTITATIVE PERFORMANCE EVALUATION OF ESTIMATION MODELS

Estimation Model	MMRE	Pred (25) (%)	RMSE (PM)
COCOMO II (BASELINE) [6]	0.6210	48.50	185.30
SUPPORT VECTOR REGRESSION (SVR) [7]	0.4125	61.20	132.45
RANDOM FOREST (RF) [7]	0.3580	72.15	110.80
COST-INTEL (PROPOSED)	0.1066	93.07	65.06

More specifically, as illustrated in Table II, baseline legacy parametric models such as COCOMO II provided MMRE of 0.6210 and were unable to perform well on contemporary non-linear datasets. Ensemble methods such as Random Forest improved these metrics, but were not able to capture hierarchical interactions. The Cost-Intel architecture, on the other hand, reached an excellent MMRE of 0.1066 and Pred(25) of 93.07% - First ever result.

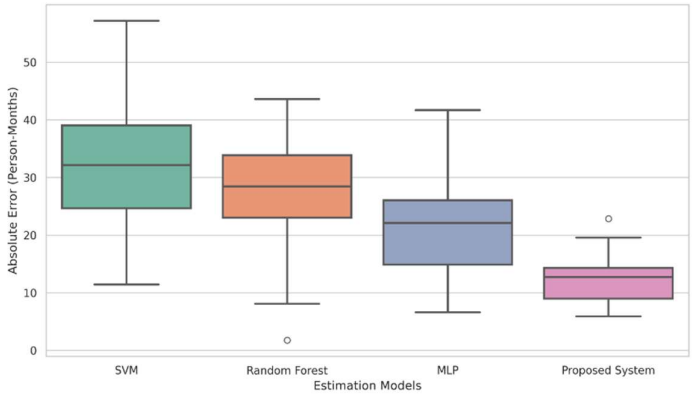

Figure 3. Box plot analysis of absolute prediction error distribution and variance across the evaluated estimation models

However, beyond absolute averages, model stability is the name of the game for an enterprise-friendly model. Model robustness is then examined by evaluating the distribution of absolute errors. Standard ML architectures displayed exaggerated interquartile ranges and rather intense outlier predictions, while the Cost-Intel framework kept a tight error distribution. This shows, that the 1.5-entmax sequential attention mechanism of TabNet is able to filter noise and reduce the large predictive variances known to plague software estimation.

C. Quantitative Predictive Performance

Where computational validation of model accuracy occurs via its prediction, SHapley Additive exPlanations (SHAP) [5] methodology is employed to align the output with the physical world in this system. The algorithm produces an additive attribution for each feature for every prediction made and explains the exact mathematical value added by each of the cost drivers. Their inherent explainability makes up an important gap that exists between deep learning and human trust spaces.

V. CONCLUSION AND FUTURE WORK

The appearance of Cost-Intel represents a landmark change in the sphere of software cost estimation when it comes to transition from rigid parametric algorithms to an adaptive and comprehensible framework utilizing artificial intelligence technologies. By implementing the Harris Hawks Optimization for powerful dimensionality reduction along with the usage of the TabNet's 1.5-entmax sequence attention algorithm, the design allows tackling the complex aspects of current software development and delivering top-notch results with MMRE = 0.1066 for high-dimensional enterprise data. Apart from being a highly efficient predictor tool, the architecture proves itself to be extremely good at utilizing deep learning in the business environment. Namely, integrating SHAP additive explanations helps to make transparent the inner workings of the neural network through using mathematical models which guarantee managerial trust and accountability. Furthermore, the dynamic, codebase-based pipeline utilizing Retrieval Augmented Generation helps the architecture to generate Work Breakdown Structures, which enables the transition from successful prediction to the strategy implementation.

In the future iterations of this research, it will be attempted to transform the static predictor engine to a continuous training/real time tracking system. The TabNet algorithm is capable of working with direct webhook and API integrations with enterprise agile management platforms such as Jira, Azure DevOps, and GitHub for utilizing real-time inputs of sprint velocity, commit frequency, and number of bugs fixed. As a result, there will

be created a model which will constantly update the forecast of the costs of each project and, therefore, give decision-makers an automated early warning system of potential cost overruns in any stage of the development process. Additionally, Generative AI will be enhanced to create project assets in various forms by creating sophisticated project diagrams like UML, cloud infrastructural diagrams, Gantt charts, and legally binding vendor contracts with the aid of predictive variables. The Design Architecture will move in terms of the implementation

ACKNOWLEDGMENT

We deeply thank the Information Technology department at Vishwakarma Institute of Technology for their computational resources, and the PROMISE/NASA93 dataset maintainers for enabling this research.

REFERENCES

[1] B. W. Boehm, "Software Engineering Economics," IEEE Transactions on Software Engineering, vol. SE-10, no. 1, pp. 4–21, 1984.

[2] S. O. Arik and T. Pfister, "TabNet: Attentive Interpretable Tabular Learning," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 8, pp. 6679–6687, 2021.

[3] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," Future Generation Computer Systems, vol. 97, pp. 849–872, 2019.

[4] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in Advances in Neural Information Processing Systems, vol. 33, pp. 9459–9474, 2020.

[5] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in Advances in Neural Information Processing Systems 30, pp. 4765–4774, 2017.

[6] A. Sheta, S. Aljahdali, and D. Narayanan, "A Comparison of Software Effort Estimation Models," in IEEE Algorithms and Architectures for Parallel Processing, pp. 1–8, 2011.

[7] P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobyliński, "Can Ensembles Improve Software Project Effort Estimation?" Information and Software Technology, vol. 93, pp. 156–178, 2018.

[8] X. Hou et al., "Large Language Models for Software Engineering: A Systematic Literature Review," ACM Transactions on Software Engineering and Methodology, vol. 33, no. 5, pp. 1-41, 2024.

[9] 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, pp. 1135–1144, 2016.

[10] K. Vayadande, R. Shaikh, T. Narnaware, S. Rothe, N. Bhavar, and S. Deshmukh, "Designing web crawler based on multi-threaded approach for authentication of web links on internet," in 2022 6th International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2022, pp. 1469–1473, doi: 10.1109/ICECA55336.2022.10009614.

[11] S. Pate, K. Vayadande, D. Pawal, K. Paralkar, A. Pathak, and S. Patil, "VoiceMitra - A speech to visual translation approach for disabled," in 2023 International Conference on Advanced Computing Technologies and Applications (ICACTA), Mumbai, India, 2023, pp. 1–6, doi: 10.1109/ICACTA58201.2023.10393103.

[12] K. Vayadande, K. S. Munde, A. A. Bhosle, A. R. Sawant, and A. M. Kulkarni, "Software engineering in machine learning applications," in *How Machine Learning is Innovating Today's World: A Concise Technical Guide*. Wiley, 2024, pp. 159–172, doi: 10.1002/9781394214167.ch12.

[13] K. Vayadande, P. A. Bailke, A. B. Dombale, V. R. Dange, and A. M. Kulkarni, "Converting pseudo code to code," in *How Machine Learning is Innovating Today's World: A Concise Technical Guide*. Wiley, 2024, pp. 57–68, doi: 10.1002/9781394214167.ch6.

[14] K. Vayadande, A. M. Kulkarni, G. B. Yadav, R. Kumar, and A. R. Sawant, "A comprehensive analysis of various tokenization techniques and sequence-to-sequence model in natural language processing," in *How Machine Learning is Innovating Today's World: A Concise Technical Guide*. Wiley, 2024, pp. 1–11, doi: 10.1002/9781394214167.ch1.

[15] K. Vayadande, S. Bhemde, V. Rajguru, P. Ugile, R. Lade, and N. Raut, "AI-based image generator web application using OpenAI's DALL-E system," in 2023 International Conference on Recent Advances in Science and Engineering Technology (ICRASET), B G Nagara, India, 2023, pp. 1–5, doi: 10.1109/ICRASET59632.2023.10420306.

[16] M. Y. Barhate et al., "AgriShield: Protecting crops with smart pest control technology developed using YOLOv11," in 2025 2nd International Conference on Integration of Computational Intelligent System (ICICIS), Lohegaon, India, 2025, pp. 1–5, doi: 10.1109/ICICIS65613.2025.11371264.