An Incremental Learning Approach for Realistic Story Point Prediction in Agile Frameworks

Mostafa Tawfiq[1], Mennatallah Ahmed, Mostafa Hani, Mohamed Alomar, Ahmed Darwish
Computer Department, Faculty of Engineering, Cairo University

***Abstract* — Story point estimation is a critical yet challenging task in agile software development. Traditional machine learning approaches often require retraining models from scratch on an entire dataset to incorporate new information, a process that is computationally expensive and impractical in dynamic project environments. This paper proposes a novel incremental learning approach that addresses this challenge. Our method begins by training a robust baseline model on previous historical projects, which is then efficiently updated as new data from ongoing projects becomes available. We demonstrate the effectiveness of this approach through experiments on both classification and regression tasks. To ensure a fair and rigorous evaluation, we compare our incremental learning model to a full retrain model, which is retrained on all initial project data combined with all previously seen batches from the new project. Our results show that incremental learning achieves performance comparable to that of the full retrain approach but with a significantly lower computational cost, making it a more realistic and scalable solution for enterprises.**

***Index Terms* — Agile software development, incremental learning, machine learning, story point estimation.**

I. Introduction

Agile software development is an approach to building software that emphasizes flexibility, customer collaboration, and delivering work in small, incremental pieces. Unlike traditional methods where everything is planned upfront.

Agile teams work in short cycles, allowing them to adapt to changes and get feedback early and often. Story point estimation is a key activity for planning and forecasting. Accurate estimations are crucial for effective project management, but they are often difficult to achieve due to the inherent uncertainty and complexity of software development tasks. Machine learning has been increasingly applied to automate and improve the accuracy of story point estimation. However, many existing approaches have limitations that hinder their practical adoption in real-world enterprise environments.

A common issue is the reliance on generic, open-source datasets for model training. These datasets are often a random collection of projects from various domains and do not reflect a specific scope of work. While useful for academic research, these datasets often fail to capture the specific nuances and development practices of a particular company. An alternative is to build project-specific models, which train on a part of the project and test on the rest, but this approach is not viable for new projects with no historical data.

This paper addresses these challenges by proposing an incremental learning approach that is both realistic and efficient. The core idea is to train a baseline model on a history of completed projects, preferably from within the same company to ensure higher accuracy. This model is then deployed to provide initial estimates for new projects. As the new project progresses and more data becomes available (e.g., at the end of each sprint), the model is incrementally updated, or "fine-tuned," to adapt to the characteristics of the ongoing project. This process is illustrated in Figure 1.

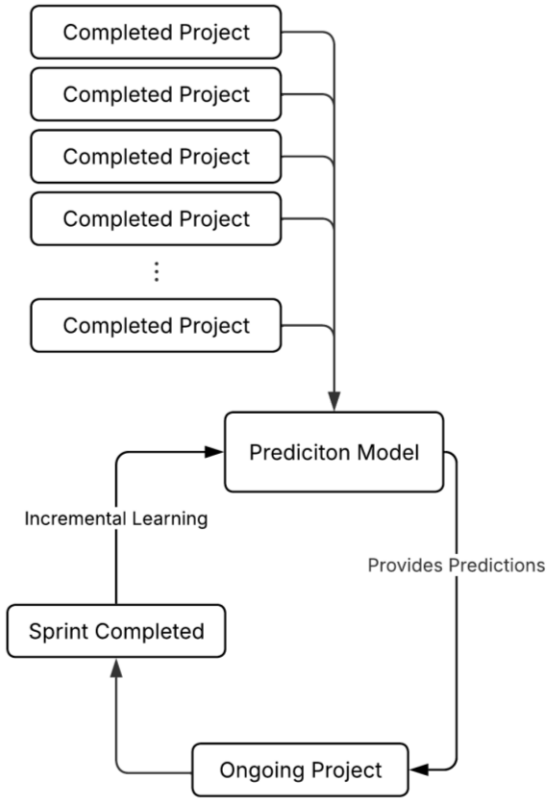

Fig. 1. The proposed incremental learning workflow. Historical data from completed projects is used for initial model training. This model provides

[1] To whom all correspondences should be addressed
mustafa.tawfiq02@eng-st.cu.edu.eg

predictions for the ongoing project. As sprints are completed, the new data is used to fine-tune the model incrementally.

We evaluate our approach using two machine learning models that support incremental updates: PassiveAggressiveClassifier for a classification-based approach to story point prediction, and SGDRegressor for a regression-based approach. We compare their performance and computational cost against two baseline methods: a static model that is never updated and a full retrain model that is retrained on all available data (historical data + data from current project) at each step.

In this paper, we demonstrate that incremental learning offers a superior trade-off between prediction accuracy and computational cost, making it a practical solution for deployment in production environments.

The remainder of this paper is organized as follows: Section 2 discusses *related work*. Section 3 details our *methodology*. Section 4 presents and analyzes *the results* of our experiments. Section 5 discusses *the implications* of our findings, and Section 6 concludes the paper.

II. Literature Survey

The application of machine learning to software engineering problems, including effort estimation, is a well-established field of research. Various techniques, from traditional regression models to more advanced deep learning architectures, have been explored for story point prediction.

A foundational study in this area was conducted by Choetkiertikul et al. [1], who introduced a deep learning model for estimating story points from the textual descriptions of issues. Their model architecture employed Long Short-Term Memory (LSTM) to generate a vector representation of the issue's text, which was then processed through Recurrent Highway Networks (RHWN) before a final regression layer predicted the story point value. Trained and evaluated on a large dataset of over 23,000 issues from 16 open-source projects, this established a strong benchmark in the field, outperforming several other machine learning techniques [2], [3], [4], [5], [6].

Following the advancements in Natural Language Processing (NLP), researchers began exploring the capabilities of Transformer-based models. Fu and Tantithamthavorn [3] introduced GPT2SP, an approach based on the GPT-2 architecture, applying a powerful large language model directly to the task of story point estimation. This marked a shift towards leveraging large, pre-trained models for software engineering tasks.

However, subsequent critical analysis has added important nuance to these findings. Replication studies conducted by Tawosi et al. [6], [7] re-evaluated the performance of GPT2SP and highlighted that its improvements over the Choetkiertikul et al. [1] model were not as pronounced as initially suggested. Their corrected results showed that GPT2SP outperformed Choetkiertikul et al. model in only a minority of cross-project scenarios, and the statistical significance of these improvements was often negligible.

A significant limitation shared by these influential models is that they are inherently static. They are designed to be trained once on a historical dataset and then used for prediction without further updates. This approach fails to address two critical realities of live software projects. First is the problem of "concept drift," where the statistical properties and nature of project tasks evolve, causing a static model's predictions to become less accurate over time. Second is the impracticality of the common alternative: retraining the entire model from scratch whenever new data becomes available. While a full retrain incorporates new information, it is computationally expensive and not a scalable solution for dynamic agile environments where sprints conclude frequently.

Furthermore, much of the existing research evaluates models in a setting that does not fully replicate a realistic use case. Methodologies often test a model on data from the same pool of projects used for training, a scenario different from making initial predictions for a completely new project where no historical data exists. Cross-project estimation remains a significant hurdle for generalization.

The concept of incremental learning, or online learning, is not new, but its application to story point estimation in a realistic enterprise context has not been thoroughly investigated.

In the realm of incremental learning, Pesala et al. [8] proposed two novel algorithms for Support Vector Machines (SVMs) that leverage backward elimination and forward selection of support vectors to enable efficient model updates in dynamic industrial settings. Their approach addresses the limitations of traditional batch SVMs by incrementally updating the model with new data chunks, significantly reducing computational time and memory usage while maintaining or improving classification accuracy compared to batch learning and existing Incremental Learning SVM (ILSVM) methods. Evaluated on 13 diverse datasets, their algorithms demonstrated robustness against concept drift, a challenge also pertinent to agile software development where task characteristics evolve over time. This work complements our proposed incremental learning framework for story point estimation, as it underscores the scalability and efficiency of incremental approaches in handling dynamic data streams. However, while Pesala et al. focus on general classification tasks, our study tailors incremental learning to the specific context of story point prediction, incorporating both classification and regression tasks to address the unique challenges of agile project management.

Our work builds upon the existing literature by not only using a cross-project evaluation methodology but also by incorporating the concept of incremental model updates to handle the stream of new data from ongoing projects.

III. METHODOLOGY

Our methodology is designed to simulate a realistic scenario where a company wants to leverage its past project data to estimate story points for a new project. We test our approach on a dataset comprising several software projects. For each experiment, we hold out one project for testing and use the remaining projects to train the initial model.

A. Dataset and Preprocessing

The model was trained and evaluated on a dataset of around 32604 issues from 40 different open-source projects [9]. The raw issue data undergoes a series of preprocessing steps to ensure data quality and prepare it for our machine learning model. These steps are executed in the following order:

1. *Combine Textual Data*: The title and description fields for each user story are concatenated into a single text field. This unified text column serves as the primary input for our model.

2. *Filter by Story Points (SP):* To standardize the effort estimation, we filter the dataset to include only issues with Fibonacci story point values of 1, 2, 3, 5, and 8. Issues with non-Fibonacci values (e.g., 4, 6, 7) or values greater than 8 are removed from the dataset.

3. *Resolve Inconsistencies:* The dataset is cleaned by identifying and excluding user stories that are textually similar but have conflicting story point assignments. This step ensures that the model is trained in consistent examples.

4. *Text Normalization:* All punctuation is removed from the consolidated text field. This helps to reduce the dimensionality of the data and focus on the meaningful words.

5. *Token Limitation:* The input text for each issue is truncated to the first 500 tokens. This prevents excessively long descriptions from disproportionately influencing the model and helps manage computational resources.

6. *Encode Story Point Values:* The filtered Fibonacci story point values are mapped to a zero-indexed numerical format for compatibility with machine learning classification algorithms. The mapping is as follows: {1→0}, {2→1}, {3→2}, {5→3}, and {8→4}.

Following these preprocessing stages, the cleaned and structured text data is then converted into numerical features using the *Term Frequency-Inverse Document Frequency (TF-IDF)* vectorizer. TF-IDF is a standard technique in natural language processing that reflects the importance of a word in a document relative to a collection of documents.

B. Experimental Setup

We frame the story point prediction problem as both a *classification* and a *regression* task. In classification the story points are treated as discrete classes. We use the *PassiveAggressiveClassifier* as our incremental model. In regression the story points are treated as continuous values. We use the *SGDRegressor* as our incremental model.

For each task, we compare the performance of our incremental model against two baselines:

1. *Static Model*: A model trained on the initial set of historical projects and never updated. This represents the naive approach of using a fixed, pre-trained model.
2. *Full Retrain Model*: A model that is retrained from scratch at the end of each batch, using all the historical data plus all the new data from the completed batches to ensure fair comparison.

For classification, we use a *Support Vector Machine (SVM)* with a linear kernel. For regression, we use *Support Vector Regressor (SVR) with RBF kernel with C=1 and gamma=scale.* This baseline represents the most accurate but also the most computationally expensive approach.

C. Cross Validation and Hyperparameter Tuning

To select the best hyperparameters for our models, we use *RandomizedSearchCV*, which is often faster than an exhaustive grid search and provides good coverage of the parameter space. A key aspect of our methodology is the use of *GroupKFold* for cross-validation. This technique ensures that all data from a particular project is assigned to either the training or the validation set within each fold, but never split across both. This prevents data leakage and provides a more realistic estimate of how the model will perform on a completely new project.

The hyperparameters tuned for each model used:

- *TF-IDF vectorizer*: max_features, ngram_range, min_df and max_df
- *PassiveAggressiveClassifier:* C, max_iter, class_weight and loss function
- *SGDRegressor*: alpha, max_iter, learning_rate, eta and regressor loss

D. Incremental Learning Process

The test project is processed in batches, simulating the sprints in an agile project. For each batch, we perform the following steps:

1. *Training*
   - The *incremental model* is updated using only the data from the current batch. This is a fast process as it doesn't require access to the entire historical dataset.
   - The *static model* is not updated.

- The *full retrain model* is retrained using all the historical data plus all the data from the batches processed so far.

2. *Evaluation*: The current models (incremental, static, and full retrain) are used to make predictions on the current batch of data. Their performance is evaluated and recorded.

This process is repeated for all batches in the test project.

E. Model Choice

The *PassiveAggressiveClassifier* was chosen for the classification task because it is an online learning algorithm that is well-suited for incremental updates. It belongs to the family of margin-based classifiers, similar to SVMs. Its "passive-aggressive" nature makes it efficient: it remains "passive" if a data point is correctly classified with a sufficient margin, and it becomes "aggressive" by updating its weights when it encounters a misclassification.

For the regression task, the *SGDRegressor* is a natural choice as it also supports incremental learning through the *partial_fit* method. It is a linear model that optimizes the loss function using *Stochastic Gradient Descent*, making it highly efficient and scalable.

IV. Results and Analysis

TABLE I
CLASSIFICATION RESULTS FOR TIMOB PROJECT

Batch	Incremental Accuracy	Static Accuracy	Full Retrain Accuracy	Inc. Time(s)	SVM Time (s)
1	0.2385	0.2176	0.2929	0.0246	605
2	0.2762	0.1632	0.2678	0.0240	609
3	0.2887	0.2343	0.3222	0.0273	621
4	0.2803	0.1883	0.2845	0.0222	622
5	0.3264	0.1841	0.3264	0.0280	629
6	0.2887	0.1715	0.3347	0.0239	653
7	0.3556	0.2008	0.3431	0.0221	665
8	0.3305	0.2176	0.2971	0.0221	668
9	0.2636	0.2218	0.3305	0.0244	701
10	0.3264	0.2176	0.3640	0.0245	710

We present the results of our experiments for both the classification and regression tasks. The experiments were run on several projects from our dataset, including *TIMOB*, *TISTUD*, *MULE*, and *XD*.

A. Classification Results

The table below shows a sample of the results for the *TIMOB* project, comparing the accuracy and training time of the incremental, static, and full retrain models.

- *Accuracy*: The incremental model's accuracy is often close to that of the full retrain model and consistently and significantly outperforms the static model.

- *Training Time*: The most striking result is the difference in training time. The incremental model updates in a fraction of a second, while the full retrain model takes several minutes for each batch. In real-world scenarios, datasets are often massive, making the full retrain approach a significant scalability problem. Attempting to retrain a model from scratch on such large volumes of data *after each development sprint* takes a prohibitive amount of time. This lengthy process makes the full retraining strategy impractical for real-world applications that require frequent and timely updates.

- *Concept Drift*: The static model's performance is erratic and often degrades over time. This is a clear sign of *concept drift*, where the statistical properties of the data change over the course of the project. The incremental model effectively handles concept drift by adapting to these changes.

TABLE II
REGRESSION RESULTS FOR TIMOB PROJECT

Batch	Incremental MAE	Full Retrain MAE	Inc. Time (s)	SVM Time (s)
1	1.8185	1.9070	0.0557	923.3
2	1.8359	1.9705	0.0578	951.0
3	1.8641	1.9698	0.0553	960.6
4	1.8407	1.9389	0.0519	974.7
5	1.8027	1.9251	0.0589	986.1
6	1.6117	1.7159	0.0602	1007.5
7	1.7246	1.8750	0.0590	1031.2
8	1.7813	1.8470	0.0521	1037.5
9	1.7294	1.7123	0.0486	1071.7
10	1.7832	1.8856	0.0563	1067.8

B. Regression Results

The table below shows the results for the regression task on the *TIMOB* project, comparing the Mean Absolute Error (MAE), and training time.

The results from the regression task mirror those from the classification task. The incremental SGDRegressor achieves a lower average MAE (better performance) score compared to the full retrain SVR model. The training time for the incremental model is negligible compared to the SVR model, which takes hours to retrain over the course of the project.

V. Discussion

The results of our experiments strongly support the case for using an incremental learning approach for story prediction due to its lower complexity resulting in hundreds of folds of faster execution which is highly needed in a realistic enterprise setting.

A. The Case for Cross-Project Learning

A key aspect of our approach is the leveraging of historical data from multiple projects. This is a good choice over project-specific models. A new project, by definition, has no data, so a project-specific model cannot be used for initial estimates. By training on a diverse set of past projects, we create a more generalized model that can provide reasonable estimates from day one of a new project. The incremental learning then allows this model to specialize in the specific characteristics of the new project as it progresses.

B. Advantages of Incremental Learning

1. *Near-Optimal Performance at a Fraction of the Cost*: While a full retrain model can sometimes achieve slightly higher accuracy, the performance gain is marginal and comes at a massive computational cost. Our results show that the incremental model can achieve comparable, and in some cases better, performance than the full retrain model, while being orders of magnitude faster.

2. *Handles Concept Drift*: Software projects are dynamic, and the nature of the work can change over time. The static model's poor performance demonstrates its inability to cope with this "concept drift." Incremental learning, by its very nature, is designed to adapt to such changes, ensuring that the model remains relevant throughout the project's lifecycle.

3. *Scalability and Practicality*: In a production environment, computational resources are often a constraint. The full retrain approach, with its high computational demands, is not a scalable solution. As the amount of historical data grows, the retraining time will become even more prohibitive. The incremental approach, on the other hand, has a constant processing time per sample, making it highly scalable and practical for real-world deployment. It is also more memory efficient as it does not require the entire dataset to be held in memory for updates.

C. Limitation and Future Work

Our study has some limitations. The experiments were conducted on a specific dataset, and the results may vary for other datasets or organizations. Future work could involve testing our approach on a wider range of datasets and exploring the use of more complex incremental learning models, such as neural networks. Another avenue for research would be to investigate more sophisticated strategies for deciding when and how to update the model, rather than updating it after every batch.

VI. Conclusion

This paper has presented a novel incremental learning approach for story point estimation that is both realistic and computationally efficient. By training a baseline model on historical data and then incrementally updating it with new data from ongoing projects, our approach overcomes the limitations of traditional methods that rely on generic datasets or are too computationally expensive for practical use.

Our experiments, covering both classification and regression tasks, demonstrate that incremental learning delivers performance that is comparable to, and in some cases better than, a full retrain approach, but with a training time that is over 99% faster. This makes it a highly scalable and practical solution for enterprises looking to improve the accuracy and efficiency of their story point estimation process. We believe that this approach represents a significant step forward in the practical application of machine learning to software project management.

References

[1] *M. Choetkiertikul, H. K. Dam, T. Tran, T. Pham, A. Ghose, and T. Menzies, "A Deep Learning Model for Estimating Story Points," IEEE Trans. Softw. Eng., vol. 45, no. 7, pp.637–656, Jul. 2019, doi:10.1109/TSE.2018.2792473*

[2] *M. Gultekin and O. Kalipsiz, "Story Point-Based Effort Estimation Model with Machine Learning Techniques," Int. J. Softw. Eng.Knowl. Eng., vol. 30, no. 01, pp. 43–66, Jan.2020, doi: 10.1142/S0218194020500035*

[3] *M. Fu and C. Tantithamthavorn, "GPT2SP: A Transformer-Based Agile Story Point Estimation Approach," IEEE Trans. SoftwEng., vol. 49, no. 2, pp. 611–625, Mar. 2022, doi: 10.1109/TSE.2022.3158252.*

[4] *V. Tawosi, A. Al-Subaihin, and F. Sarro, "Investigating the Effectiveness of Clustering for Story Point Estimation," in 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), Mar. 2022, pp. 827–838. doi:10.1109/SANER53432.2022.00101.*

[5] *B.Marapelli, A. Carie, and S. M. N. Islam, "RNN-CNN MODEL:A Bi-directional Long Short-Term Memory Deep Learning Network For Story Point Estimation," in 2020 5th International Conference on Innovative Technologies in Intelligent Systems and Industrial Applications (CITISIA), Sydney, Australia: IEEE, Nov. 2020, pp. 1–7. doi: 10.1109/CITISIA50690.2020.9371770.*

[6] *V. Tawosi, R. Moussa, and F. Sarro, "Agile Effort Estimation: Have We Solved the Problem Yet? Insights From a Second Replication Study (GPT2SP Replication Report)," Sep. 01, 2022, arXiv: arXiv:2209.00437. doi:10.48550/arXiv.2209.00437.*

[7] *V. Tawosi, R. Moussa, and F. Sarro, "Agile Effort Estimation: Have We Solved the Problem Yet? Insights From a Replication Study," IEEE Trans. Softw. Eng., vol. 49, no.4, pp. 2677–2697, Dec. 2022, doi:10.1109/TSE.2022.3228739.*

[8] *V. Pesala, A. K. Kalakanti, T. Paul, K. Ueno, A. Kesarwani and H. G. S. P. Bugata, "Incremental Learning of SVM Using Backward Elimination and Forward Selection of Support Vectors," 2019 International Conference on Applied Machine Learning (ICAML), Bhubaneswar, India, 2019, pp. 9-14, doi: 10.1109/ICAML48257.2019.00010.*

[9] *V. Tawosi, A. Al-Subaihin, R. Moussa, and F. Sarro. "A Versatile Dataset of Agile Open-Source Software Projects." https://arxiv.org/abs/2202.00979, 2022.*