Enhancing Software User Interface Testing Through Few Shot Deep Learning: A Novel Approach for Automated Accuracy and Usability Evaluation

Aris Puji Widodo[1*], Adi Wibowo[2], Kabul Kurniawan[3]
Dept. of Computer Science, Universitas Diponegoro, Semarang, Indonesia[1, 2]
Dept. Information Systems and Operations Management, Vienna University of Economics and Business (WU), Vienna, Austria[3]

Abstract*—Traditional user interface (UI) testing methods in software development are time-consuming and prone to human error, requiring more efficient and accurate approaches. Moreover, deep learning requires extensive data training to develop accurate automated UI software testing. This paper proposes an efficient and accurate method for automating UI software testing using Deep learning with training data limitations. We propose a novel deep learning-based framework suitable for UI element analysis in data-scarce situations, focusing on Few-shot learning. Our framework initiates with several robust feature extraction modules that employ and compare sophisticated encoder models to be adept at capturing complex patterns from a sparse dataset. The methodology employs the Enrico and UI screen mistake datasets, overcoming training data limitations. Utilizing encoder models, including CNN, VGG-16, ResNet-50, MobileNet-V3, and EfficientNet-B1, the EfficientNet-B1 model excelled in the setting of Few-Shot learning with five-shot with an average accuracy of 76.05%. Our proposed model's accuracy was improved and compared to the state-of-the-art method. Our findings demonstrate the effectiveness of few-shot learning in UI screen classification, setting new benchmarks in software testing and usability evaluation, particularly in limited data scenarios.

***Keywords*—Deep learning; efficientnet; few-shot; software testing; UI screen classification**

I. Introduction

Evaluating a User Interface (UI) in computer science requires expertise and partnership to assess its practical benefits for the intended users [1]. Improved UI design enhances user satisfaction and contributes to increased retention and revenue [2]. Fundamental principles in UI design involve exploring solutions, identifying specific attributes, and actively engaging users in the design process [3]. Despite these principles, the manual testing of UIs remains cumbersome due to the hands-on verification of requirements [4]. Nowadays, computer vision based on neural networks is used to advance UI understanding and address this challenge [5].

This study's main contribution is applying the few-shot learning approach, a notable advancement in computer vision based on neural networks, to effectively classify UI screens. This approach becomes particularly relevant when dealing with limited dataset availability and a diverse range of classes. Our implementation involves applying the few-shot learning technique to the Enrico dataset, a dataset of UI screens, and curating a novel dataset encompassing ten classes that represent common UI mistakes. This strategic combination of few-shot learning and creating a dedicated dataset allows us to explore and enhance UI screen classification in a data-driven manner, especially in scenarios with limited data.

A key advantage of few-shot learning is its ability to handle the constraints of limited sample sizes. This is crucial in fields like UI testing, where obtaining large, diverse datasets can be challenging. Few-shot learning techniques are designed to learn effectively from a few examples, making them ideal for situations where data scarcity is a critical issue. By leveraging this approach, our study aims to demonstrate how advanced computer vision based on neural network (see Fig. 1) techniques can overcome traditional data limitations, opening new avenues for efficient and accurate UI testing.

II. Related Work

Deep learning models have brought numerous benefits to the software development industry, significantly enhancing various tasks, including UI testing [6]. Deep Residual Networks (ResNet) have improved considerably image classification, surpassing previous methods on ImageNet. The successful implementation of ResNet signifies a crucial advancement in utilizing deep learning for image classification tasks [7]. The success of deep learning frameworks extends to automating test case generation and repairing unstable tests in functional UI testing [8].

The application of deep learning in UI understanding, particularly addressing Graphical User Interface (GUI) complexity, has been demonstrated in studies such as OwlEye, which achieved an 85% precision and 84% recall. It uncovers previously-undetected UI display issues in popular Android apps [9]. The use of an enhanced dataset from Rico further illustrates the effectiveness of deep learning in UI topic modeling, achieving notable accuracy in screenshot representation [10]. Challenges in machine learning datasets for UI modeling, mainly due to manual collection limitations, have led to the introduction of large datasets like WebUI, emphasizing the need for enhanced visual UI understanding in domains with limited labeled data [11].

*Corresponding Author.

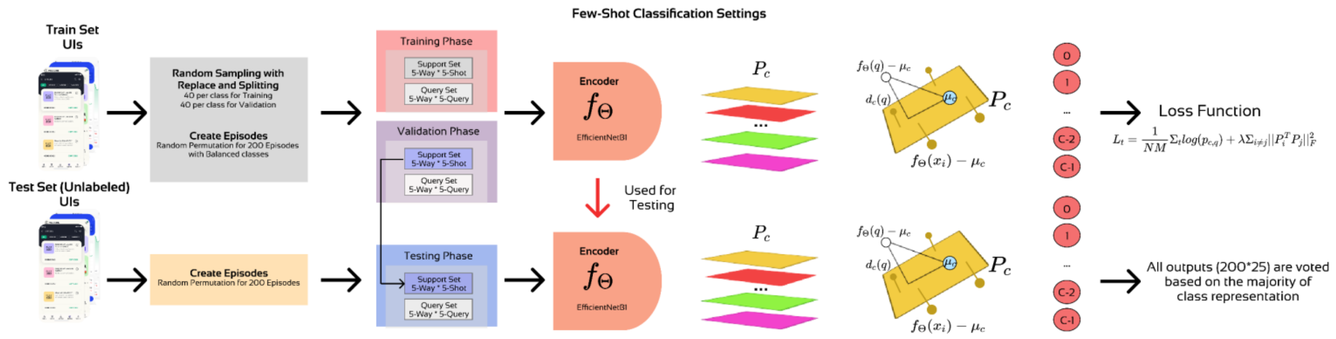
Few-Shot Classification Settings
Train Set UIs
Test Set (Unlabeled) UIs
Random Sampling with Replace and Splitting
40 per class for Training
40 per class for Validation
Create Episodes
Random Permutation for 200 Episodes with Balanced classes
Create Episodes
Random Permutation for 200 Episodes
Training Phase
Validation Phase
Testing Phase
Support Set 5-Way * 5-Shot
Query Set 5-Way * 5-Query
Encoder
f_Θ
EfficientNetB1
Used for Testing
P_c
$f_\Theta(q) - \mu_c$
$d_c(q)$
μ_c
$f_\Theta(x_i) - \mu_c$
0
1
C-2
C-1
Loss Function
$L_t = \frac{1}{NM}\Sigma_t log(p_{c,q}) + \lambda\Sigma_{i\neq j}\|P_i^T P_j\|_F^2$
All outputs (200*25) are voted based on the majority of class representation

Fig. 1. Proposed neural network framework.

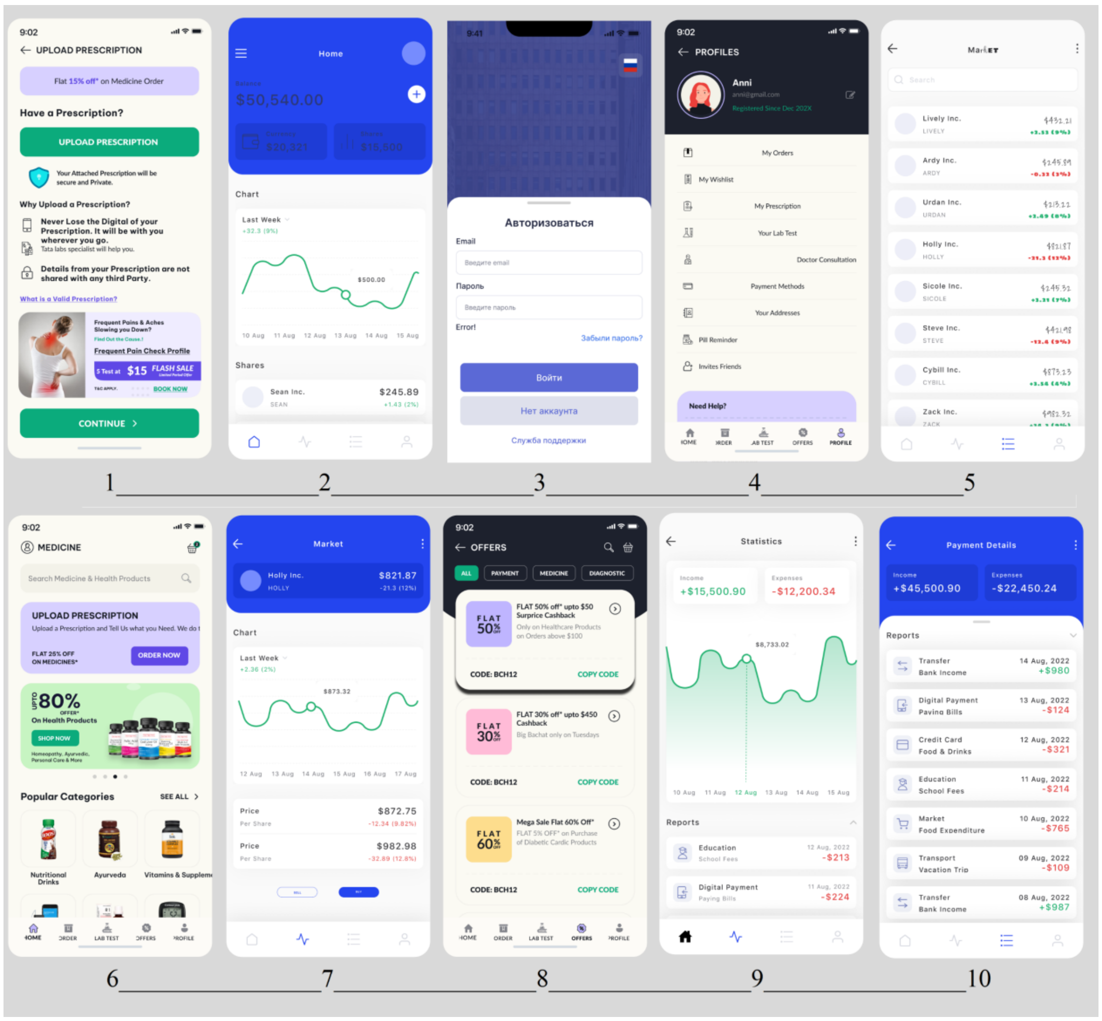

Fig. 2. Sample of the mistake dataset.

Furthermore, researchers thoroughly analyzed current few-shot image classification techniques, categorizing algorithms into transfer learning, meta-learning, data augmentation, and multimodal approaches to address challenges posed by limited sample data [12]. For example, the study introduced GenericConv, a new few-shot learning model for scene classification. Evaluation of benchmark datasets showed that GenericConv successfully addressed imperfections in previous attempts, outperforming benchmark models on three datasets [13]. Another study focused on Few-shot Class Incremental Learning (FSCIL) in real-world applications, introducing the Efficient Prototype Replay and Calibration (EPRC) method. EPRC significantly improved classification performance on CIFAR-100 and miniImageNet compared to mainstream FSCIL methods [14]. In medical imaging, a groundbreaking study proposed a novel few-shot learning method for classifying heart diseases, achieving high segmentation performance and a remarkable 92% accuracy in classifying cardiomyopathy patient groups, even without additional clinical features [15].

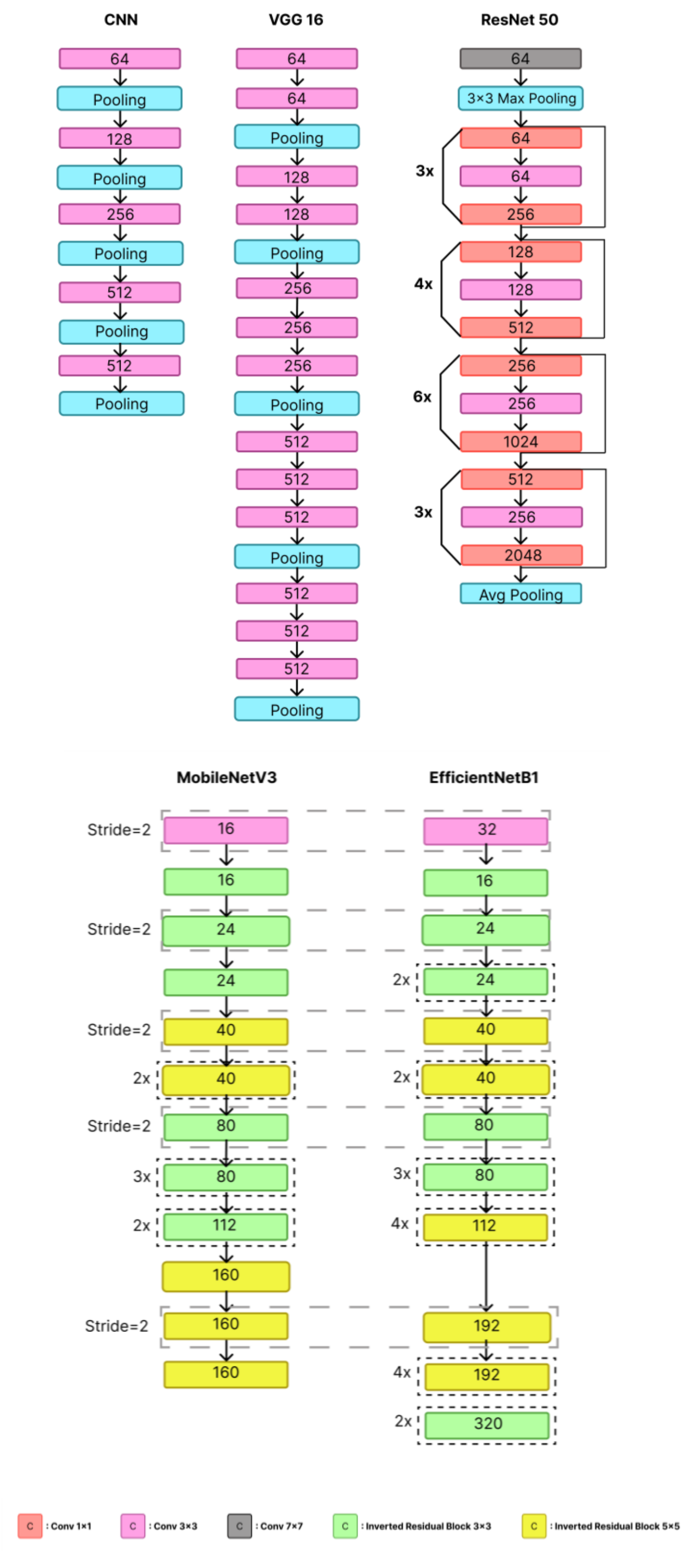

Fig. 3. Encoder Architecture

III. Proposed Method

A. Neural Network Architecture

Our method utilizes a neural network architecture comprising an encoder and an adaptive classifier for few-shot learning. We compare commonly used encoders, including basic CNN, VGG-16, ResNet-50, MobileNet-V3, and EfficientNet-B1, as illustrated in Fig. 3. This experiment marks a novel chapter by leveraging these robust architectures to train the Few-Shot Classification model. The output of these encoders is fed into an adaptive classifier model configured for a few-shot setting.

1) Basic CNN consists of alternating layers of convolution and pooling. Convolutional layers detect local features in the image, while pooling layers reduce the spatial size of the representation, decreasing the number of parameters and computation in the network, which helps prevent overfitting.

2) VGG-16 [16], developed by the Visual Geometry Group at the University of Oxford, is a widely used convolutional neural network (CNN) architecture for image classification. With 16 layers, including 13 convolutional layers and three fully connected layers, VGG-16 excels in feature extraction. The convolutional layers employ small, local filters, capturing hierarchical features as input progresses. The subsequent pooling layers enhance computational efficiency and translation invariance. The final fully connected layers process high-level features, mapping them to specific classes for classification. While VGG-16's deep and intuitive design proves effective in various computer vision tasks, its computational cost limits real-time applications despite being a foundational model for advanced CNN architectures.

3) ResNet-50 [17], developed by Microsoft Research, is a deep convolutional neural network (CNN) renowned for overcoming challenges in training intense networks. Utilizing the concept of residual learning, it introduces skip connections to mitigate the vanishing gradient problem, facilitating the training of deeper networks. Its architecture, featuring residual blocks with skip connections, allows the extraction of intricate features. The initial convolutional layers detect low-level features, while residual blocks, incorporating multiple convolutional layers, use skip connections to learn residual functions. Maximum-pooling layers aid computational efficiency, and fully connected layers map the known features for classification. ResNet-50's innovative architecture enables the training of intense networks, proving highly effective in various computer vision tasks, notably image classification and object recognition.

4) MobileNet-V3 [18], developed by Google, is a lightweight convolutional neural network (CNN) tailored for efficient and accurate computer vision tasks on mobile and edge devices. The model prioritizes high performance while minimizing computational resources, introducing innovative features like inverted residuals and linear bottlenecks to optimize efficiency. Inverted residuals utilize lightweight depthwise separable convolutions, reducing parameters and computational load, while linear bottlenecks balance representational capacity and computational cost. Integration of Squeeze and Excitation (SE) blocks enhances feature recalibration, focusing on crucial channels. MobileNetV3's multiple building blocks collectively form a streamlined and efficient architecture suitable for resource-constrained environments. This design makes it ideal for applications on devices with limited computational resources, offering a commendable trade-off between accuracy and model size in various computer vision tasks, particularly on mobile and edge platforms.

5) EfficientNet-B1 [19], a member of the EfficientNet family developed by Google, is a convolutional neural network (CNN) designed to attain high accuracy with minimal computational complexity. Characterized by balanced scaling of depth, width, and resolution, it optimizes performance across these dimensions. Employing a compound scaling method, EfficientNet-B1 ensures efficient feature extraction at various abstraction levels. Increasing depth, width, and resolution enhances the model's capacity to capture complex and diverse features. Compounding scaling achieves a harmonious balance, maximizing computational resources and performance. This enables EfficientNet-B1 to achieve state-of-the-art accuracy on various computer vision tasks while maintaining a low computational cost. Its adaptability in handling different scaling factors makes it particularly advantageous for resource-constrained environments, striking an optimal balance between accuracy and model size.

B. Few Short Classification

The classification model was built using the few-shot learning approach to deal with restricted data. In itself the meta-learning mode, few-shot terminology is used to train many samples during the training phase; an episode of $\mathcal{T}_i$ was made up of two types of sets: support set $S = \{(x_{1,1}, c_{1,1}), \ldots, (x_{N,K}, c_{N,K})\}$ and query set $Q = \{q_1, \ldots, q_{N\times M}\}$. The number of S and Q was restricted for each iteration based on N-way, which specifies the number of classes, and K-short or M-query, which denotes the number of samples in each class. The few-shot model is divided into two parts: encoder and classifier.

In this work, a dynamic classifier based on an adaptive subspace [20] was applied. The subspace was used for classification, as well as the shortest distance between the data points and their projection into the subspace. A collection of samples encoded by may be stated as $\tilde{X}_c = [f_\theta(x_{c,1}) - \mu_c, \ldots, f_\theta(x_{c,k}) - \mu_c]$, where $\mu_c = \frac{1}{K}\sum_{x_i \in X_c} f_\theta(x_i)$. For instance, q is the query set, and the subspace classifier computation is as follows:

$$d_c = -||(I - M_c)(f_\theta(q) - \mu_c)||^2 \quad (1)$$

where, $M_C = P_C P_C^T$ and μ_c signify the offset between the data point and the subspace, P_C is the truncated matrix of matrix C_C with an orthogonal basis for linear subspace spanning $\mathbb{x}_C = \{f_\theta(x_i); y_i = c\}$ $(hence, B_C^T B_C = I)$.

The chance of a query falling into class c may be calculated using the SoftMax function, which is written as follows:

$$p_{c,q} = p(c|\boldsymbol{q}) = \frac{\exp(d_c(\boldsymbol{q}))}{\sum_{c'} \exp(d_c(\boldsymbol{q}))} \quad (2)$$

Backpropagation through singular value decomposition can be used to minimize the negative log from $p_{c,q}$.

During training, the projection metric on Grassmannian geometry was utilized as a discriminative approach to maximize the margin between two subspaces P_i and P_j, and it is defined as follows:

$$\delta_p^2(P_i, P_j) = \left|\left|P_i P_i^T - P_i P_j^T\right|\right|_F^2 = 2n - 2\left|\left|P_i^T P_j\right|\right|_F^2 \quad (3)$$

the projection metric was maximized by reducing $||P_i^T P_j||_F^2$ and then developing a loss function as stated in Eq. (4); $\mathcal{L}_t$ may then be used to update θ.

$$\mathcal{L}_t = -\frac{1}{NM}\sum_c \log(p_{c,q}) + \lambda \sum_{i\neq j} ||P_i^T P_j||_F^2 \quad (4)$$

IV. Empirical Research Methodology

A. Dataset

1) Enrico dataset [10], is a carefully selected subset originating from Rico, an extensive mobile app dataset. Enrico, a short form for Enhanced Rico, comprises 1,460 UIs categorized into 20 specific design topics. Each category delineates particular UI features, contributing to a detailed comprehension of mobile app design. These topics encompass diverse aspects such as Bare (largely unused area), Dialer (number entry), Camera (camera functionality), Chat (chat functionality), Editor (text/image editing), Form (form-filling functionality), Gallery (grid-like layout with images), List (elements organized in a column), Login (input fields for logging), Maps (geographic display), MediaPlayer (music or video player), Menu (items listed in an overlay or aside), Modal (popup-like window), News (snippets list: image, title, text), Other (everything else, considered as a rejection class), Profile (information on a user profile or product), Search (search engine functionality), Settings (controls to change app settings), Terms (terms and conditions of service), and Tutorial (onboarding screen).

2) The Mistake Dataset is a dataset proposed by this paper as shown in Fig. 2. This dataset consists of 200 user interfaces (UI) categorized into 10 specific design topics. Each category describes specific UI features that contribute to a detailed understanding of mobile device UI design. These topics encompass various aspects such as pointless inconsistency design, inapproriate use of shadows, lack of text hierarchy, bad iconnography, unaligned elements, low contrast, poor typography choices, tiny touch targets, text overlap, and error message clarity. For detailed explanations regarding the 10 design topics, (see Table I).

TABLE I. DESCRIPTION MISTAKE DATASET

No	UI Component	Description
1	Pointless inconsistency design	Identifying inconsistencies in layout across different screens or resolutions
2	Inapproriate use of shadows	Pay attention to the depth of the shadow, they should create a sense of realism and hierarchy
3	Lack of text hierarchy	The text format must have a contract between each text style and the title style
4	Bad iconnography	Ensuring icons are correctly displayed and recognizable
5	Unaligned elements	Detecting misaligned text, buttons, image or other
6	Low contrast	Identifying poor contrast between text and background, affecting readability
7	Poor typography choices	Font selection, font size, spacing and alignment must be taken into account to facilitate readability
8	Tiny touch targets	The size of the touch target must be taken into account with the size of the screen
9	Text overlap	Spotting instances where text overlaps with UI elements
10	Error message clarity	Ensuring that error message are clear, concise, and appropriately placed

B. UI Understanding and Methodology

The initial technologies developed for understanding app user interfaces operate at either the application or screen level, aiming to inspire new designs by exploring existing relevant ones. For instance, [21] creates a searchable gallery of UI element ideas based on app pictures. Enrico [10] draws inspiration from extensive datasets that are too vast to browse effectively, showcasing the evolving landscape of UI exploration.

To enhance design analysis and automation, screen type or functionality classification from a screenshot proves beneficial. Enrico [10], a dataset with 1460 samples (a subset of Rico [22]), categorizes each screenshot into one of 20 design categories. However, the limited dataset size poses challenges for training deep learning classification models. Despite having a vast online dataset, it lacks screen type annotations, preventing the utilization of the pre-training method employed for element recognition.

In the methods section of our research, we present a strategic solution to overcome the limitations of small datasets in training deep learning models for UI screen classification. By integrating a Few-Shot Learning Approach for UI Screen Classification, we enable our models to perform effectively with the Enrico [10] dataset, which comprises a modest collection of 1460 samples. This approach is tailored to efficiently learn from a constrained number of data points, thus allowing accurate classification of each screenshot into one of the proposed 20 design categories. The few-shot learning technique is pivotal in our methodology, as it addresses the challenge of dataset scarcity, a common obstacle in the realm of UI design analysis.

Further augmenting our methodological framework, we have developed a dataset named “Mistake of UI Screen”, encompassing 10 classes of prevalent UI design errors, alongside the use of the Enrico dataset. This dataset is crafted to refine our models' ability to not only discern various UI elements but also to detect and classify typical design flaws systematically. The development of this dataset, alongside the application of few-shot learning, constitutes a robust approach to enhancing the precision of screen classification, thereby contributing significantly to the field of UI design and evaluation.

V. EXPERIMENTAL RESULT

In this study, a few-shot learning approach was applied using Enrico dataset with 20 class and Mistake dataset with 10 class. Randomly selects 40 data per class and splits them to 50% training and 50% validation phase with balanced class distribution. During testing phase, we randomly sampled query images along with the support set from the validation phase, repeating this process twenty-five times and ultimately applying majority voting in a few-shot setting. Settings used were five-way for all phases, five-shot on support and query sets, and 200 episodes for each epoch. Training iteration using 10 epochs for Enrico dataset referring to original paper and 25 epochs for Mistake dataset with learning rate of 1 x 10^{-3} optimized by Adam and lambda of 0.03. Few-shot model ran on an i7 processor with RTX 2060 SUPER.

A. Model Implementation

1) Model and shot setting in enrico dataset: Our assessment of model performance involved a sequence of experiments using the validation subset of the Enrico dataset. Table II compiles the accuracy figures for five models: CNN, VGG-16, ResNet-50, MobileNet-V3, and EfficientNet-B1, across five separate experiments. The table outlines each model's individual experiment results, along with their respective average accuracies and standard deviations. EfficientNet-B1 emerged as the top performer, boasting an average accuracy of 76.05% with a standard deviation of 1.1618%. The findings suggest that models, particularly those with depth and designed for efficiency that leverage pre-trained networks, have the potential to enhance generalizability.

The comparison across models were proposed to further investigated the influence of training variation on the performance of EfficientNet-B1. Table III shows the accuracy results for EfficientNet-B1 when trained with varying shots numbers on validation subset of the Enrico dataset. These conditions ranged from 1 to 5-shot training, with each tested over five experimental runs. The data indicates that increasing the number of shots enhances both the accuracy and stability of the model's performance. Definitely, the five-shot trained EfficientNet-B1 achieved a highest average accuracy, prominence that a greater number of shots correlate with improved model performance.

TABLE II. ACCURACY RESULT FOR DIFFERENT MODELS APPLIED TO THE ENRICO DATASET ON THE VALIDATION SUBSET

Model	Experiment 1	Experiment 2	Experiment 3	Experiment 4	Experiment 5	Average	Standard Deviation
CNN	0.4020	0.4219	0.4108	0.3976	0.4008	0.4066	0.009843
VGG-16	0.3120	0.4020	0.3506	0.3013	0.3702	0.3472	0.041501
ResNet-50	0.4800	0.4480	0.4373	0.5164	0.5004	0.4764	0.033632
MobileNet-V3	0.7012	0.7230	0.7012	0.7148	0.7119	0.7104	0.009349
EfficientNet-B1	0.7650	0.7432	0.7506	0.7703	0.7732	0.7605	0.011618

TABLE III. ACCURACY RESULT FOR EFFICIENTNET-B1 WITH VARIOUS SHOTS APPLIED TO THE ENRICO DATASET ON THE VALIDATION SET

Number of Shots	Experiment 1	Experiment 2	Experiment 3	Experiment 4	Experiment 5	Average	Standard Deviation
1-shot	0.5078	0.6800	0.5509	0.5509	0.5860	0.5751	0.064853
2-shot	0.6800	0.6530	0.6230	0.6002	0.6230	0.6358	0.031002
3-shot	0.6800	0.7100	0.6980	0.6800	0.7098	0.6956	0.015012
4-shot	0.7011	0.7130	0.7266	0.6980	0.7100	0.7097	0.011263
5-shot	0.7650	0.7432	0.7506	0.7703	0.7732	0.7605	0.011618

TABLE IV. ACCURACY RESULT FOR EFFICIENTNET-B1 WITH 5- SHOT APPLIED TO THE MISTAKE DATASET ON THE VALIDATION SET

Experiment	Accuracy
Experiment 1	0.3737
Experiment 2	0.4800
Experiment 3	0.3820
Experiment 4	0.4670
Experiment 5	0.4302
Average	0.4266
Standard Deviation	0.0431

2) Model and shot setting in mistake dataset: The assessment of the performance EfficientNet-B1 model with five-shot by testing were applied on the validation subset of the Mistake Dataset. The results of this investigation are shows in Table IV about details the model's accuracy across five experiments. These experiments were designed to evaluate the model's ability to predict accurately under different conditions. The overall performance is summarized by the mean accuracy, calculated to be 42.66%. Additionally, the standard deviation of the accuracy, at 0.0431, indicates a relatively small variability in the model's performance throughout the trials, suggesting a stable accuracy profile for the EfficientNet-B1 on this dataset.

B. Comparison with other Method

Table V shows a brief accuracy comparison of various models on Enrico dataset. It contrasts the performance of a model trained on the Screenshot data from the Enrico dataset, achieving 75.8% accuracy, with that of the VGG-16 and Noisy ResNet-50 models trained on the Enrico dataset, which have accuracies of 47.4% and 46.5% respectively. Our adaptation of the EfficientNet-B1 model also performed notably well, with an accuracy close to the top-performing Screenshot model at 76.1%. These outcomes demonstrate the strong performance of our EfficientNet-B1 model, which nearly matches the leading model and substantially exceeds the performance of established architectures like VGG-16 and Noisy ResNet-50. EfficientNet-B1 achieves higher accuracy than CNN, VGG-16, ResNet-50, and MobileNet-V3 due to its optimized balance of depth, width, and resolution in the network architecture. It is designed using a compound scaling method to scale these three dimensions efficiently and in harmony. This results in more effective and efficient use of computational resources and better performance on limited data, as in Few Shot learning scenarios, making it particularly suitable for the complex task of UI screen classification.

TABLE V. ACCURACY COMPARISON ON OTHER METHOD

Model Configuration	Accuracy
Screenshot [10]	75.8%
VGG-16 [23]	47.4%
Noisy ResNet-50 [23]	46.5%
EfficientNet-B1 (our)	**76.1%**

VI. DISCUSSION

In the discussion section of our analysis, we delve into the substances and insights derived from our research findings. We initiate by examining our model's impact on the Enrico and Mistake datasets, scrutinizing how the few-shot learning approach adapts to each dataset's unique characteristics. The transition was used to discuss the broader implications of our model on UI screen classification, highlighting the advancements and the potential it unlocks for future applications. Finally, we address the limitations encountered during our research and offer suggestions for future studies to build upon our work.

A. Model Impact on each Dataset

The application of our neural network model to the Enrico dataset demonstrated remarkable adaptability, as evidenced by the high accuracy rates achieved across diverse UI screen types. This success is attributed to the model's ability to learn from limited examples, a testament to the efficacy of few-shot learning. Conversely, when applied to the Mistake dataset, which contained a different array of UI screen errors, the model faced distinct challenges, reflecting the nuances and complexity of classifying a wider variety of mistakes. These observations underscore the need for tailored approaches when dealing with datasets of varying natures.

B. Improved UI Screen Classification

Our model represents a significant step forward in the realm of UI screen classification. By leveraging few-shot learning, we have shown that it is possible to achieve high levels of precision with limited training data, a scenario common in real-world settings. This advancement is particularly promising for the development sector, where rapid and accurate UI assessment can streamline the design process, reduce costs, and enhance the end-user experience.

C. Limitation and Suggestion

Our primary constraint was the size and diversity of the datasets, which may affect the model's ability to generalize across a broader range of UI designs. For future work, we suggest expanding the datasets to include a more varied set of UI screens and errors. Further, investigating other few-shot learning configurations and incorporating user feedback in the training loop could refine the model's performance and applicability.

VII. CONCLUSION

In this paper, we propose an efficient and accurate method for automating user interface (UI) testing using Deep learning with training data limitations. This research proposes a novel deep learning-based framework that marks a pivotal advancement in user interface (UI) testing, demonstrating the powerful capabilities of few-shot learning in UI screen classification. The framework initiates with several robust feature extraction modules that employ and compare sophisticated encoder models (CNN, VGG-16, ResNet-50, MobileNet-V3, and EfficientNet-B1) to be adept at capturing complex patterns from a sparse dataset. EfficientNet-B1 achieves higher accuracy than CNN, VGG-16, ResNet-50, and MobileNet-V3 due to its optimized balance of depth, width, and resolution in the network architecture. It is designed using a compound scaling method to scale these three dimensions efficiently and in harmony. This results in more effective and efficient use of computational resources and better performance on limited data, as in Few Shots learning scenarios, making it particularly suitable for the complex task of UI screen classification. Moreover, our proposed model's accuracy was improved and compared to the state-of-the-art method using the Enrico dataset. Our utilization of the Enrico and Mistake datasets confirmed the model's ability to accurately classify a broad spectrum of UI screens with limited training data, illuminating its proficiency in detecting and categorizing intricate UI errors. For the rapidly evolving domain of software development, our model offers a scalable and efficient solution to the perennial challenge of UI testing. This is crucial for the development sector, where quick, reliable UI assessments can significantly expedite the design process, cut costs, and elevate the end-user experience. Moreover, this study paves the way for future research endeavors. The challenges we encountered due to our datasets limited size and diversity underline the need for more expansive and varied data in further investigations. Such efforts could refine the model's applicability and accuracy, integrating user feedback and diverse learning configurations into the development process.

ACKNOWLEDGMENT

This work was supported by the Research for International Publication (RKI) NonAPBN Universitas Diponegoro Indonesia under Grant 609-78/UN7.D2/PP/VIII/2023.

REFERENCES

[1] N. Samrgandi, “User Interface Design & Evaluation of Mobile Applications User Interface Design & Evaluation of Mobile Applications,” no. February, 2021.

[2] K. Edson et al., “An Evaluation Framework for User Experience Using Eye Tracking , Mouse Tracking , Keyboard Input , and Artificial Intelligence : A Case Study,” International Journal of Human–Computer Interaction, vol. 00, no. 00, pp. 1–15, 2021, doi: 10.1080/10447318.2021.1960092.

[3] M. Bakaev, M. Speicher, J. Jagow, S. Heil, and M. Gaedke, “We Don’t Need No Real Users?! Surveying the Adoption of User-less Automation Tools by UI Design Practitioners,” Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 13362 LNCS, no. July, pp. 406–414, 2022, doi: 10.1007/978-3-031-09917-5_28.

[4] Z. Khaliq, D. A. Khan, and S. U. Farooq, “Using deep learning for selenium web UI functional tests: A case-study with e-commerce applications,” Engineering Applications of Artificial Intelligence, vol. 117, no. August 2022, p. 105446, 2023, doi: 10.1016/j.engappai.2022.105446.

[5] G. Ang and E. P. Lim, “Learning User Interface Semantics from Heterogeneous Networks with Multimodal and Positional Attributes,” International Conference on Intelligent User Interfaces, Proceedings IUI, pp. 433–446, 2022, doi: 10.1145/3490099.3511143.

[6] F. H. Alshammari, “Trends in Intelligent and AI-Based Software Engineering Processes: A Deep Learning-Based Software Process Model Recommendation Method,” Computational Intelligence and Neuroscience, vol. 2022, 2022, doi: 10.1155/2022/1960684.

[7] M. Shafiq and Z. Gu, “Deep Residual Learning for Image Recognition: A Survey,” Applied Sciences (Switzerland), vol. 12, no. 18, pp. 1–43, 2022, doi: 10.3390/app12188972.

[8] Z. Khaliq, S. U. Farooq, and D. A. Khan, “A deep learning-based automated framework for functional User Interface testing,” Information and Software Technology, vol. 150, no. December 2021, p. 106969, 2022, doi: 10.1016/j.infsof.2022.106969.

[9] Z. Liu, C. Chen, J. Wang, Y. Huang, J. Hu, and Q. Wang, “Owl Eyes: Spotting UI Display Issues via Visual Understanding,” Proceedings - 2020 35th IEEE/ACM International Conference on Automated Software Engineering, ASE 2020, pp. 398–409, 2020, doi: 10.1145/3324884.3416547.

[10] L. A. Leiva, A. Hota, and A. Oulasvirta, “Enrico: A Dataset for Topic Modeling of Mobile UI Designs,” Extended Abstracts - 22nd International Conference on Human-Computer Interaction with Mobile Devices and Services: Expanding the Horizon of Mobile Interaction, MobileHCI 2020, 2020, doi: 10.1145/3406324.3410710.

[11] J. Wu, S. Wang, S. Shen, Y. H. Peng, J. Nichols, and J. P. Bigham, “WebUI: A Dataset for Enhancing Visual UI Understanding with Web Semantics,” Conference on Human Factors in Computing Systems - Proceedings, 2023, doi: 10.1145/3544548.3581158.

[12] Y. Liu, H. Zhang, W. Zhang, G. Lu, Q. Tian, and N. Ling, “Few-Shot Image Classification : Current Status and Research Trends,” 2022.

[13] M. Soudy and Y. M. Afify, “GenericConv : A Generic Model for Image Scene Classification Using Few-Shot Learning,” pp. 1–13, 2022.

[14] W. Zhang and X. Gu, "Few Shot Class Incremental Learning via Efficient Prototype," 2023.

[15] A. Wibowo et al., "Cardiac Disease Classification Using Two-Dimensional Thickness and Few-Shot Learning Based on Magnetic Resonance Imaging Image Segmentation," Journal of Imaging, vol. 8, no. 7, 2022, doi: 10.3390/jimaging8070194.

[16] M. Ye et al., "A Lightweight Model of VGG-16 for Remote Sensing Image Classification," IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 14, pp. 6916–6922, 2021, doi: 10.1109/JSTARS.2021.3090085.

[17] B. Koonce, "ResNet 50," Convolutional Neural Networks with Swift for Tensorflow, pp. 63–72, 2021, doi: 10.1007/978-1-4842-6168-2_6.

[18] J. Huang et al., "BM-Net: CNN-Based MobileNet-V3 and Bilinear Structure for Breast Cancer Detection in Whole Slide Images," Bioengineering, vol. 9, no. 6, 2022, doi: 10.3390/bioengineering9060 261.

[19] C. Wang and Y. Li, "Motion Prediction for Autonomous Vehicles Based on EfficientNet-B1," 2022 4th International Conference on Communications, Information System and Computer Engineering, CISCE 2022, pp. 648–651, 2022, doi: 10.1109/CISCE55963.2022 .9851007.

[20] C. Simon, P. Koniusz, R. Nock, and M. Harandi, "Adaptive subspaces for few-shot learning," Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 4135–4144, 2020, doi: 10.1109/CVPR42600.2020.00419.

[21] C. Chen, S. Feng, Z. Xing, L. Liu, S. Zhao, and J. Wang, "Gallery D.C.: Design search and knowledge discovery through auto-created GUI component gallery," Proceedings of the ACM on Human-Computer Interaction, vol. 3, no. CSCW, 2019, doi: 10.1145/3359282.

[22] B. Deka et al., "Rico: A mobile app dataset for building data-driven design applications," UIST 2017 - Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology, pp. 845–854, 2017, doi: 10.1145/3126594.3126651.

[23] J. Wu, S. Wang, S. Shen, Y. H. Peng, J. Nichols, and J. P. Bigham, WebUI: A Dataset for Enhancing Visual UI Understanding with Web Semantics, vol. 1, no. 1. Association for Computing Machinery, 2023.