Wireframe Prototyping using Machine Learning

1st Bhumika Chuphal
Dept. of CSE
Graphic Era Hill University
Dehradun, India
bhumikachuphal125@gmail.com

2nd Himanshu Dangwal
Dept. of CSE
Graphic Era Hill University
Dehradun, India
himanshudangwal99@gmail.com

3rd Ishmita Negi
Dept. of CSE
Graphic Era Hill University
Dehradun, India
ismitanegi78@gmail.com

4th Abhishek Rana
Dept. of CSE
Graphic Era Hill University
Dehradun, India
ar145501@gmail.com

***Abstract*—To convert a user interface drawing into wireframe code using App Inventor, a popular block-based programming environment, is a typical requirement for mobile application development by end users or in computing education. We offer the Sketch2aia technique, which automates this difficult and time-consuming job. Deep learning is used by Sketch2aia to automatically generate the App Inventor code for the wireframe after identifying the most common user interface elements and their locations on a hand-drawn sketch to create an intermediate representation of the user interface. The method produces wire-frames that closely resemble the sketches in terms of visual likeness, according to the findings of a preliminary user evaluation, and it achieves an average user interface component categorization accuracy of 87,72%. The method has been put into practise as a web tool and may be used to successfully and efficiently support end-user development of mobile applications as well as the teaching of user interface design in K–12 classrooms.**

Keywords— Machine learning, mobile applications, and graphical user interfaces

1. INTRODUCTION

A successful and profitable mobile application must offer a useful, effective, and enjoyable experience with an appealing user interface (UI), as there are millions of mobile applications available today, from social media to digital entertainment to medical treatments. As a result, user interface design is crucial to the product strategy, highlighting its significance in the development of mobile apps . It is not simply a small part of the creation of the app. Wireframe prototyping is typically used in the development of user interfaces (UIs), beginning with the creative process of creating sketches, which are straightforward hand-drawn representations. They are an efficient visual medium for exchanging ideas, having discussions, and quickly comparing various options.Wireframes, which depict the interface layout and structure without visual design elements like colours, graphics, etc., are developed from sketches to define the visual hierarchy. The visual design is then added to the wireframe after it has been built and refined, transforming it into a high-fidelity prototype . Once the design is complete, it is put into practise to create the finished product. These phases can be repeated numerous times as an iterative prototype process during the creation of an interface, necessitating rework anytime changes are made. As a tedious, time-consuming, and error-prone operation, the creation of code from UI prototypes has a great potential for automation in this context.

While designers typically use graphic editors like Photoshop or Illustrator to create user interfaces, a wide range of design tools, such as Sketch, Figma, Marvel, or Adobe XD, have developed into all-in-one tools for designing, prototyping, and testing, though they still need the UI design to be coded. Modern IDEs, on the other hand, feature robust interactive drag-and-drop based builders for UI code, like those found in Eclipse, Visual Studio, or Android Studio. However, even with the technologies now in use, translating wire-frames or sketches into code still requires manually rebuilding user interfaces . App Inventor (MIT 2020), a well-known visual block-based programming environment that has allowed more than 50K people worldwide to create more than 34K mobile applications, is another example of this.

Due to the fact that a growing number of mobile applications are being created by individuals with knowledge in other fields as well as young people as part of K–12 computing education, App Inventor thus offers a crucial programming environment. The automatic generation of code for user interfaces (UIs) of websites and mobile applications that often utilize Machine Learning (ML) methodologies is therefore being addressed by solutions . These programs automatically translate front-end code or code representations from hand-drawn wire-frames or sketches. This automation streamlines the UI development process, saves time and effort, and aids in avoiding unintentional errors . Additionally, despite the fact that there are now a number of methods for automatically creating the code for mobile applications from hand-drawn designs, there is currently no method for producing App Inventor code . We employ a deep learning approach to build an intermediate representation of the UI that is used to automatically generate App Inventor code in order to automatically detect UI elements in sketches. The online tool can be used to help end-user development and the teaching of UI design as part of computing education, saving time on the manual construction of wire-frames.

2. LITERATURE REVIEW

The main motive behind making this project is to assist users to quickly bring into code their creativity in an easier and better way.It will save plenty of time and since in the technical era times is proportional to money therefore it will be very efficient for companies as their designers have to spent less time sitting around and just attempt to bring their creativity into reality via code[1].

There exist solutions for web UIs as well as mobile applications, focusing more on Android than iOS applications. Yet, so far, no solution specifically for App Inventor apps and or computing education is currently available. Most approaches are based on sketches as input, recognizing the importance of the creative process of sketching as a first step in prototyping the UI in contrast to approaches that aim at complete automation of the UI design process. However, a shortcoming of several approaches is their rather explorative way of focusing on only a very small number of UI components.[2]

Research concerning images of UIs is further complicated due to the unavailability of large datasets that also include sketches. For example, the popular RICO dataset (Deka et al. 2017) being one of the largest datasets on user interfaces of mobile apps with information on over 9.3k Android apps, focuses on the representation of visual, textual, structural, and interactive properties based on screenshots as well as metadata, does not provide, for example, sketches. Thus, considerable effort needs to be spent on the creation of specific datasets. [3]

Adopting diverse approaches, some studies captured UI screenshots by crawling the web or app stores. Others generated UI images or sketches synthetically, which may limit the image quality as well as authenticity Yet, this issue and its potential impact on performance and validity that is not further discussed in most cases. An exception is Robinson (2019), who reports a reduction of the performance of the deep learning approach when applied to real sketches, emphasizing, thus, the importance of a great variety in sketches to allow the deep learning approach to better generalize to unseen sketching styles.[4]

For object detection, a large variety of solutions have been created ranging from classic Machine Learning approaches to recent convolutional neural networks (CNNs), and diverse combinations of them. In order to be able
to detect context-related information, such as nested UI widgets, some approaches also employ sequence- and structure-analyzing recurrent convolutional neural network-based techniques.[5]

3. PROPOSED MODEL

3.1 Summary

In this model, we have created a large datset manually. This data set contains images of different types of sketches of layout like texts, buttons, images and other types of wire-frames. [1]

To get better and more reliable results, data pre-processing was applied to the data. For this purpose, the PyTorch library is used here which includes common transformations like normalization. It is first resized and then convert the rgb image to the gray scale image. Normalization is then applied to find the mean and standard deviation. After doing all this to each image of dataset, data augumentation is also applied.[2]

Data is splitted into training and testing phase with the ratio of 70 : 30 . Training more images increases the accuracy of the model and classification result is likely to be correct in a very efficient way.[3]

A pre-trained AlexNet is used which is specialize tin the extraction of the features and also for the classification. AlexNet is already trained by large dataset which is called ImageNet.[4]

After completion of the training process, the model will take unlabeled images for the classification. In testing phase, this model will take image and then compare and as the output, it will give the digital prototype of the wieframes that can be used by the user for designing the Graphical user interface. [5]

3.2 Methodology

We created an online tool for the automatic production of code for wire-frames in App Inventor based on sketches using a multi-method approach based on the findings of previous work on many types of user interfaces. We created a deep learning model for the detection of UI elements in sketches based on the findings of a comprehensive literature review.

Figure 1 shows the conversion of sketch into digital prototype and Figure 2 shows the different steps included in this model.

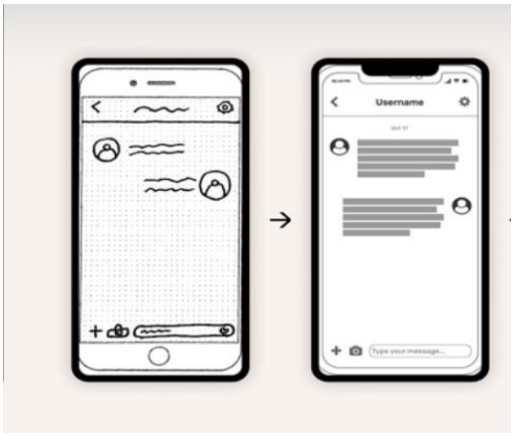

Figure 1: Conversion of wireframe to digital prototype

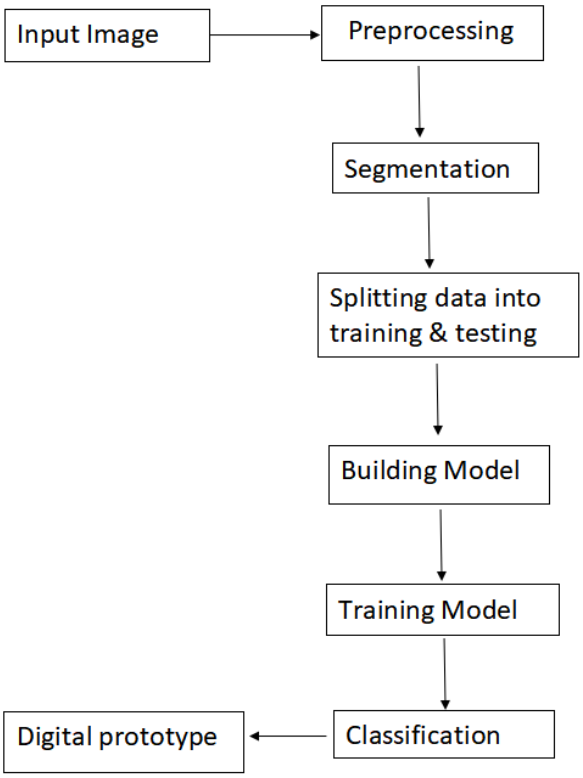

Figure 2: Various Steps Included

3.2.1 Requirement Analysis

The primary goal of the model and its target attributes are determined during this phase. Additionally, it specifies the problem by characterizing the inputs and desired results. The below table shows the record of the elements included in the dataset.

Table 1: Dataset Record

Screen	100%
Label	90%
Button	76%
Textbox	85%
Checkbox	45%
Image	36%
Dropdown	10%
Sliders	1%

3.2.2 Data Management

When gathering data, we looked for readily accessible datasets. This involves choosing from a pool of general datasets that can be used to pre-train a model (such as ImageNet for image classification tasks) as well as specialized datasets that can be used for transfer learning to train a particular model. We gathered precise data by taking UI screenshots of App Inventor projects and creating the corresponding designs because datasets on the user interfaces of App Inventor apps were unavailable.

Example of the taken UI screenshot is shown in Figure 2. The process of validating, cleaning, and conditioning the data involves dealing with issues such as duplication, errors, missing values, normalization, data type conversions, and others. Using the labelme tool and supervised learning, labels indicating UI elements in sketches were manually assigned . The dataset is divided into a training set for the model to be trained and a test set for an objective performance evaluation of the selected model on unlabeled data. model education.

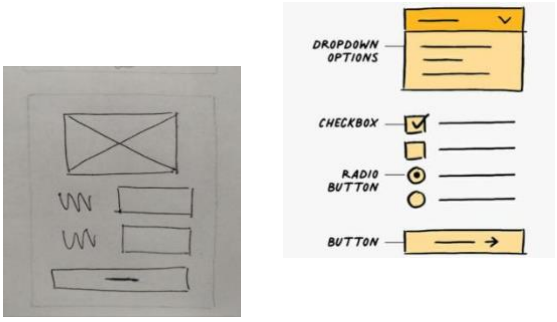

Figure 3: UI Screenshots

3.2.3 Data Loading

The loading of the dataset is the next step. The crime dataset is loaded using the Data World website. Many times every day, the datasets are updated. It shows how the dataset provides comprehensive details of the accidents and crimes that occurred in a specific longitude and latitude of a particular location.

3.2.4 Model Learning

An appropriate deep learning framework and machine learning model that has been demonstrated to be successful for a related problem or domain, as well as the volume and structure of the data, have been selected for the model learning phase. Using a transfer learning method, we train the input layer and the final, fully connected output layers on our dataset while initially maintaining the pre-trained internal feature representation structure of the network. The network's output layers are then modified to work with our evaluation metrics. Until the network does not get any better, this is done. We unfreeze the internal characteristics of the transfer-trained model after transfer learning so that all of its layers can continue to learn. The internal characteristics are then properly tuned to our data during a subsequent training step known as fine-tuning, using the same dataset of the particular domain. During the fine-tuning procedure, a collection of hyperparameters (HYPO), particularly learning momentum and learning rates, are chosen and dynamically optimised. In this case, we trained numerous related model variations with various HYPO and compared the outcomes.

3.2.5 Detection of UI components

The output of the detection of UI components in sketches is an intermediate representation, in the form of a list of detected objects, the respective confidence levels of classification, and their positions in the image (coordinates of the central point, width, and height)

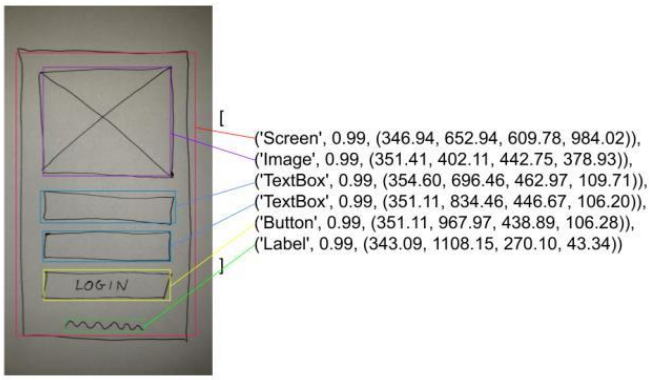

Figure 4. Output after detection

The first step to generate the App Inventor wireframe code is to eliminate possible overlapping of components. Therefore, each pair of elements is checked if their area of overlap is greater than 50% of the area of the smallest element between them, eliminating the ones with a lower confidence index.

3.2.6 Wireframe code generation

As a result, a preliminary illustration of the user interface elements and their arrangement as shown in the sketches is produced. The corresponding App Inventor code for the wireframe representation is produced based on this representation. As a result, we created a software module using an incremental and iterative software development approach. A web application has been created with the results. This comprised modelling, implementation, testing, and analysis of the context and needs. Integration tests were run following this initial implementation, and depending on the outcomes of these tests, the required changes and enhancements were put into place and tested.
. We conducted a preliminary evaluation utilising the Goal Question Metric (GQM) technique to identify the aim of the evaluation and to systematically divide it into quality characteristics in order to assess the quality of the approach. To gather information on the perceived visual similarity of the created wireframe and the sketch as well as the perceived usefulness and usability of the method, we conducted a user study in accordance with the required characteristics. Descriptive statistics were used to examine the data that was gathered. Further observations made during the user study were taken into account when interpreting the results.

3.2.7 Evaluation

The results are displayed in the last stage of this model. For this stage, there are two prequisited: First, the code against wireframe. Second: The fully working front-end similar to input hand-drawn sketch.

With this method, wire-frames in App Inventor may be generated automatically from sketches.The UI components and layout in the submitted photographs of UI sketches for a mobile application are recognized, creating an interim representation of the UI. Based on this illustration, the wireframe's App Inventor code is automatically generated and made available to the user for download, where it may then be imported and further developed into a mobile application.

4. RESULT

These findings offer a preliminary demonstration of the EasySketch approach's ability to precisely identify objects in mobile application sketches and automatically produce App Inventor code from visually comparable wire-frames.

Given that the object detection performance evaluation received an accuracy of 88%, it is clear that the method can accurately identify the UI elements in sketches. When comparing these results to the state of the art, the approach's quality is clearly clear because it shows performance values that are higher than those published by analogous publications that assessed the accuracy and precision of their approaches.

5. CONCLUSION

This research study proposes a method that leads users in the direction of the conversion of hand-drawn sketches into the graphical user interface. This proposed a method for using App inventor to automatically prototype the user interfaces of mobile applications. According to a preliminary analysis of the method, it is capable of correctly identifying and categorizing user interface elements and their locations in a design as well as producing App Inventor code for wire-frames that visually resemble the sketches. The technique may support the user interface design process as part of the creation of mobile apps, according to preliminary feedback. With this method, wire-frames in App Inventor may be generated automatically from sketches.The UI components and layout in the submitted photographs of UI sketches for a mobile application are recognized, creating an interim representation of the UI. Based on this illustration, the wireframe's App Inventor code is automatically generated and made available to the user for download, where it may then be imported and further developed into a mobile application.

Some suggestions, like supporting the creation of web interfaces as well, textual recognition for the automatic generation of the text of user interface elements, and support for the subsequent stages of user interface design, such as the choice of a colour scheme as part of the visual design, were outside the purview of our current research.

6. FUTURE WORK

In the future, we intend to broaden the scope of the approach's evaluation and improve it by adding suggestions for user interface best practices and illustrations.

7. REFERENCES

1) Venkatesh Gauri ShanSuleri, S., Pandian, V. P. S., Shishkovets, S., Jarke, M.: Eve: A Sketch-based Software Prototyping Workbench. In: Conference on Human Factors in Computing Systems, Glasgow, Scotland, UK, pp. 1–6 (2019)

2) Yun, Y.: Detection of GUI elements on sketch images using object detector based on deep neural networks. In: 6th Int. Conference on Green and Human Information Technology, Chiang Mai, Thailand (2018)

3) Zou, Z., Shi, Z., Guo, Y., Ye, J.: Object Detection in 20 Years: A Survey. arXiv:1905.05055v2 [cs.CV] (2019)

4) Moran, K., Bernal-Cárdenas, C., Curcio, M., Bonett, R., Poshyvanyk, D.: Machine Learning-Based Prototyping of Graphical User Interfaces for Mobile Apps, IEEE Transactions on Software Engineering, 46(2), pp. 196-221 (2018)

5) Kim, B., Park, S., Won, T., Heo, J., Kim, B.: Deep-Learning based Web UI Automatic Programming. In: Int. Conference on Research in Adaptive and Convergent Systems, Honolulu, USA (2018)

6) Jain, V., Agrawal, P., Banga, S., Kapoor, R., Gulyani, S.: Sketch2Code: Transformation of Sketches to UI in Real-time Using Deep Neural Network. arXiv:1910.08930 [cs.CV] (2019)

7) Huang, F., Canny, J. F., J. Nichols, J.: Swire: Sketch-based User Interface Retrieval. In: CHI Conference on Human Factors in Computing Systems, ACM, New York, NY, USA, Paper 104, pp. 1–10 (2019)

8) Beltramelli, T.: Hack your design sprint: wireframes to prototype in under 5 minutes, Medium, (2019) https://uxdesign.cc/hack you-design-sprints-wireframes-to-prototype-in-under-5-minutes-b7b95c8b2aa2

9) Aşıroğlu, B., Mete, B. R., Yıldız, E., Nalçakan, Y., Sezen, A., Dağtekin, M., T. Ensari, T.: Automatic HTML Code Generation from Mock-up Images Using Machine Learning Techniques. In: Scientific Meeting on Electrical-Electronics and Biomedical Engineering and Computer Science, pp. 1-4 (2019)

10) Deka, B., Huang, Z., Franzen, C., Hibschman, J., Afergan, D., Li, Y., Nichols, J., Kumar, R.: Rico: A Mobile App Dataset for Building Data-Driven Design Applications. In: 30th Annual Symposium on User Interface Software and Technology, ACM, New York, NY, USA, pp. 845–854 (2017)

11) Ge, X.: Android GUI search using hand-drawn sketches. In: 41st Int. Conference on Software Engineering, IEEE Press, pp. 141–143 (2019)