Jurnal Ilmu Komputer dan Informasi (Journal of Computer Science and Information)
13/1 (2020), 35-45. DOI: http://dx:doi:org/10:21609/jiki:v13i1:845

VISUAL RECOGNITION OF GRAPHICAL USER INTERFACE COMPONENTS USING DEEP LEARNING TECHNIQUE

Agyl A. Rahmadi* and Aris Sudaryanto

Informatics Department, Faculty of Engineering, Universitas 17 Agustus 1945 Surabaya,
Jl. Semolowaru No. 45, Surabaya, 60118, Indonesia

*E-mail: agyl.rahmadi@untag-sby.ac.id

Abstract

Graphical User Interface (GUI) building in software development is a process which ideally need to go through several steps. Those steps in the process start from idea or rough sketch of the GUI, then refined into visual design, implemented in coding or prototype, and finally evaluated for its function and usability to discover design problem and to get feedback from users. Those steps repeated until the GUI considered satisfactory. Computer vision technique has been researched and developed to make the process faster and easier; for example generating code for implementation, or automatic GUI testing using component images. But among those techniques, there are still few for usability testing purpose. This *preliminary research* attempted to make the foundation for usability testing using computer vision technique by built dataset which has images of various GUI components, and used the dataset in deep learning experiment for GUI components visual recognition. The experiment results showed deep learning technique suitable for the intended task using transfer learning as preferable method, with accuracy achieved at 95% for recognition of two different types of component, between 80 – 94% for two similar types of component, and above 70% for six different types of GUI components.

Keywords: *User Interface, Usability Testing, GUI, Computer Vision, Deep Learning*

Abstrak

Pembuatan antarmuka grafis (GUI) dalam perangkat lunak adalah sebuah proses yang idealnya perlu melewati beberapa tahapan. Tahapan-tahapan tersebut dimulai dari ide atau sketsa kasar dari GUI, dikembangkan menjadi desain visual, diimplementasikan dengan pengodean atau purwarupa, hingga akhirnya dilakukan evaluasi fungsi dan kebergunaan untuk mencari tahu masalah desain dan mendapatkan umpan balik dari pengguna. Tahapan-tahapan tersebut dilakukan berulang kali sampai GUI dianggap memuaskan. Teknik visi komputer telah diteliti dan dikembangkan untuk mempercepat dan mempermudah proses tersebut; seperti menghasilkan kode program implementasi dari desain visual, atau pengujian GUI secara otomatis berdasarkan citra komponen. Tapi dari teknik-teknik yang ada, belum banyak yang terkait uji kebergunaan. *Penelitian awal* ini mencoba meletakkan fondasi untuk melakukan uji kebergunaan menggunakan teknik penglihatan komputer dengan membangun dataset dari berbagai komponen-komponen GUI, dan menggunakan dataset tersebut untuk eksperimen pengenalan visual dari komponen GUI menggunakan teknik deep learning. Hasil eksperimen menunjukkan teknik deep learning cocok untuk tujuan yang diinginkan menggunakan metode transfer lerning, dengan tingkat akurasi yang dicapai 95% untuk pengenalan dua jenis komponen yang berbeda, antara 80-94% untuk dua jenis komponen yang serupa, dan di atas 70% untuk enam jenis komponen GUI yang berbeda.

Kata Kunci: *Antarmuka Pengguna, Uji Kebergunaan, GUI, Computer Vision, Deep Learning*

1. Introduction

In an interactive software, there are user interfaces (UIs) which used by user to interact with the system. The most common form of UI is graphical user interface (GUI) because of its visual nature which allows direct manipulation of the software. In many software, GUI plays important role to ease the use of it by utilizing visual design and human cognitive aspects such as correct use of colors, or how human comfortable with visual structure when reading contents. Development of GUI which does not pay attention to those two aspects, could lead to human errors, business loss, and even death of a patient [1]. One of the important factors which define whether a GUI could easily use by user is called usability.

Usability is a quality attribute which measure how easy to learn, how efficient to use, or how pleasant a UI are [2]. In a UI design, one of principles commonly used is Eight Golden Rules of Interface Design [3]. One of the rule from the principle is "seek universal usability" where a system should have plasticity. In related to visual aspect, it means facilitating transformation of contents or the view medium. It is already implemented in a web design technique called responsive web design [4]. It helps a website to be able to be viewed from various screen devices, and its contents should be realigned according to the screen –so it improves the usability of the website.

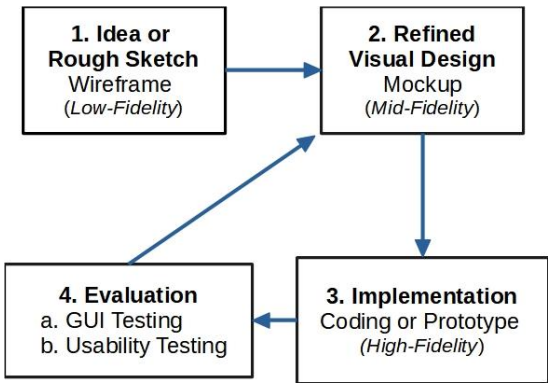

Figure 1. Simplified stages of GUI development, fidelity levels, and its deliverable

With the importance of it, User-Centered Design (UCD) method [5] is now commonly employed in UI design and development which could ensure good usability of it and does not lead to fatal problem. UCD dictate itself as iterative design process where a UI should not be developed once and for all, but in a phased and iterative process. In related to GUI, iterative UI design could lead to better quality even in redesign process [6]. In practice, visual aspect of a GUI develops in stages of increasing fidelity [7]–[9] which could be simplified and summarized as seen in figure 1. Iteratively, steps 2 until 4 from the figure above repeated until the GUI is considered satisfactory, which means satisfy, or at the minimum it should be acceptable by the users or business requirements, and does not have serious problem whether in functionality or usability.

While GUI with good usability is very important, developing it still consume development resources, especially time, with all of those steps. Even though the ideal process of the development is repeating the mockup to evaluation stage until the result is acceptable, it is often neglected because of limited resources, management interests, and communication process [10]. Furthermore, integrating GUI and functionality process often limits how it can be wholly redesigned and reimplemented.

2. Literature Review

To address those problems, previous works that utilized computer vision, or A.I in general had done in order to sped up or to aid the process of GUI development. The works tried to improve the steps from figure 1 which either sketch, mockup, implementation, or evaluation.

Computational Vision Approaches

One of earlier works that utilized computer vision technique was done by Riedl & Amant [11] and Gibss et al [12] where they made system called SegMan and Lens respectively. SegMan was an attempt to make an automatic exploration of a GUI using pixel groups segmentation based on cognitive model. While Lens was an improvement from SegMan where it added higher level of user interface abstraction such as interfaces or units, and capable ran on multiple OSes. Both system was for evaluation purpose even though addressed different problem; where SegMan towards automatic exploration of UI, while Lens more on understanding of UI (structure and classification) and was part of bigger system called Visual Total Access System (VisualTAS) that aims to help blind user in using GUI. Another works for evaluation purposes done by Chang et al. [13]. They made a system for helping tester to automate GUI testing process (evaluation step). The system called Sikuli Test enabled testers to write script test based on images of GUI components, or even generate the visual

script test after manually run the test once. It was platform agnostic and could be used to test desktop, web, or mobile applications.

Usability Factors Evaluation

For usability evaluation where the UI evaluated as a whole, Koch & Oulasvirta [14] did a work for evaluating layout perception of a UI based on algorithmic representation of gestalt psychology principles –which closely tied to human visual perception such as proximity [15]. Oulasvirta et al. [16] also made an online service called Aalto Interface Metrics (AIM) where it provides 17 different metrics evaluation using a page of UI. Those metrics based on many different researches which has validated result on user perception and attention towards GUI (e.g color blindness, white space). Works by Liu et al. [17] similarly built a system that generates semantic information of UIs from mobile apps screenshots. From the images, view hierarchy is defined, then it is segmented by colors, and finally annotated semantically based on GUI components classification. The system utilized Rico dataset [18] and focused on method development.

Deep Learning & CNN

Some of the recent works use deep learning technique in their researches. Deep learning [19] itself is a class of machine learning technique that allows raw data with multiple level of abstraction to be processed without any feature engineering such as done in conventional machine learning technique. For example of multilevel abstractions is an image of object which has edges, motif of edges, edges' motifs that form shapes, and shapes that from objects. Those layer of features can be learned using deep learning by the machine using a streamlined learning process using only the images as input data. In the conventional method, those layers need to be processed one by one which is very challenging.

In its implementation, deep learning technique commonly use one type of deep neural network which proven success in many practical computer vision applications called convolutional neural networks (CNN) or ConvNets [19]. CNN differ itself with traditional neural networks (NN) by able to accept input data that come in form of multiple arrays; in term of images, they are mostly 2-dimensional array of pixels with three channels of colors. CNN enable deep learning to accept raw input data and process it in its deep multilayer architecture which is a series of stages. Each of the stages composed of different types of layers.

Type of layers in CNN has its own function. The commonly used layers in CNN are: convolutional (conv), pooling (pool), and fully connected (dense). In computer vision applications, conv layer producing feature maps of an image from a convolution operation. That feature maps goes through additional operation called rectified linear units (ReLu) to introduce non-linearity operation to conform with real world data which often is non-linear. Concretely, ReLu make the feature maps with negative values zero by applying $max(0,x)$ function [20]. The feature maps then goes through pool layer to downsize it but still retains the most important feature information. The last layer is dense layer where its act is similar to multi layer perceptron (MLP) in traditional NN, to calculate the classification based on feature maps and usually using softmax function. In practice, CNN architecture could consist of multiple sequence of conv, pool, ReLu, and dense layer which repeated many times before producing final output.

Deep Learning Technique

A system that used deep learning done by Fernandez & Deja [21] where they did a work in usability evaluation for automatic websites heuristic evaluation using CNN. They built the dataset based on heuristic score provided by participants for various websites screenshots. Lu et al. [22] also did similar work where they collect dataset of software GUI and classified them into positive (good) or negative (bad) categories based on its page layout, comfort, and brightness. Another utilization of deep convolutional network done by Hassan et al. [23] where they made GUI components detection system from an image of UI mockup, so it could enhance GUI development. Nguyen et al. [24] also did similar work in utilizing deep learning where they proposed DeepUI system that use recurrent neural networks (RNN) to learn UI design pattern, and generative adversarial network (GAN) to generate visual design from a wireframe. Both later works developed for mobile apps development, and supports step 1 until 3 of process from Fig. 1.

In related to those steps, the mentioned Rico dataset done by Deka et al. [18] were aimed at supporting data-driven design including UI layout and code generation. Previously Nguyen & Csallner [25] made REMAUI that capable in generating mobile apps UI based on its mockup. The system processes the mockup image using computer vision technique, generates source code, and deploy it to mobile phone. Next work by Beltramelli [26] also support GUI generation for three platforms: iOS, Android, and Web-based. In tune with other works, pix2code system also use image of UI mockup for generating UI code. It was capable of 77% accuracy in generating GUI. Another work in mobile GUI development called

ReDraw by Moran et al [27] was built in order to help the process of mockup to prototype. It was able to reach 91% of component classification and generate acceptable code while maintain visual affinity with the mockup. Similar system also built by Chen et al. [28] where their work was able to generate GUI skeleton code from a mockup design. While most works in aiding GUI development start from mockup, there are works by Robinson [29] and Kim et al. [30] where they made system capable in generating web UI from sketch or wireframe stage.

Uncharted Map in Usability Evaluation

In all those mentioned works, almost half of them deal with first to third step in GUI development: sketching – mockup – implementation. It is understandable as those steps take major portion of GUI development time. While the rest deal with evaluation process of GUI, it is still divided into two problems: GUI Testing and Usability Testing. Research related to GUI testing in general proposed method for automatic functional testing, whereas works related to usability testing attempted to automate usability factors evaluation which usually done by experts. Even though many works already done for automatic usability evaluation, it still leaves rooms for exploration in that topic such as typography and readability, forms usability, or even autonomous user for usability testing. Hence, this paper attempted to also explore the possibilities in the topic.

The most common way to evaluate usability automatically is by using computer vision technique as done by many mentioned researches. One of common method is by detecting GUI components inside a UI, classify them, and process them further such as segment the components by colors and annotate them semantically [17], segment the layout based on gestalt perception [14], or even generate GUI code implementation [24] [25] [26] [27] [28] [30].

Dataset from Previous Works

Some of those results achieved by train the system using UIs dataset in deep learning process. The dataset required in the process to teach the system for learning and understanding about GUI. That is why some of previous works was for specific platforms because it depends on the dataset. Some of dataset openly available such as REMAUI [25], pix2code [26], ReDraw [27], and Rico [18]. REMAUI, ReDraw, and Rico dataset mainly consist of mobile apps screenshots mined from official apps marketplace such as Google Play Store or Apple App Store; while pix2code has synthesized GUI screenshots for different kind of app platforms. In ReDraw dataset, there are images data of various GUI components which are similar to this study intended dataset – but not satisfy all of planned classification. The dataset designed to have various standard GUI components such as text field, button, etc. In ReDraw, the dataset classification is based on Android system UI building term where it need to be reclassified to conform our needs. Many of them are also from mobile specific UI whereas this research wanted to build general GUI componets dataset. Therefore ReDraw dataset does not included in our dataset

This *preliminary research* attempted to layout foundation for a computer vision system which able to evaluate usability of a UI automatically. From previous works' dataset, all of them consists of many whole apps screenshot; which utilized by detecting the GUI components first, then followed by other step; in other words their dataset approaches started from bigger picture and processed to get detail of the UI (top-down approach). This work tried different approach by building GUI components images, and start building system to bigger picture in future works (bottom-up). The dataset itself is used to make basic computer vision application which is recognition –where the machine could classify various GUI components correctly.

At least there are two contributions in this research: 1) Initial GUI Component Images Dataset, and 2) Proof of Concept of GUI Recognition system using the dataset and deep learning technique. The technique [19] was chosen because of its capability in computer vision technique used in previous works.

3. Methods

In general, there were three steps that are done in this research: Dataset Building, Deep Learning Experiment, and Result Analysis which each respective step contains several processes. Dataset building consist of four processes: defining dataset sources, building from the sources, cleaning dataset, prepare data for experiment. After that, the dataset is used as input in the experiment in which there were four important phases: data preparation, data augmentation, setting CNN models, and training & validation. For the result, there were visualization of training and validation data from the experiment, and the analysis of the visualization for drawing conclusions. The diagram in figure 2 shows the steps done in this research.

Defining Dataset Sources

The first step in this research was building the dataset required to be used in deep learning process. The purpose is to make a system that able to discern differences between GUI components, so it is important to define which platform of UI that become the target of the system. In the long run, it is expected the system could be applied to any platform, but in this work images of GUI components from some CSS framework and design systems that is available online was used. Web-based UI is chosen because in recent years it became more prominent in software development that some non-web platforms use web-based technology such HTML, CSS, and Javascript to be used in apps development (e.g. Universal Windows Platform/UWP, Electron.js).

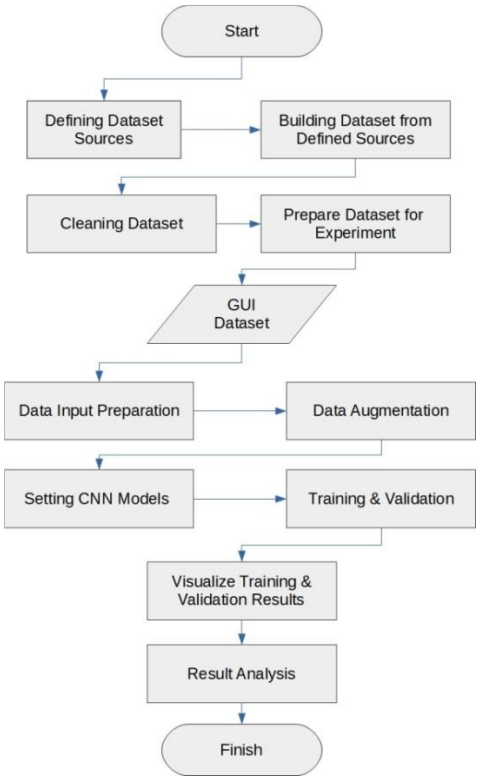

Figure 2. Steps in the research method

The dataset in this research taken from the following CSS framework: Bootstrap, Foundation, Pure CSS, Semantic UI, UI Kit, Bulma, Tailwind CSS, Materialize CSS, Picnic CSS, Paper CSS, Primer CSS; and for the design system, it was utilized the images from the following: Carbon Design System, United States Web Design System, Ant Design System, Mozilla Protocol, Blueprint, and Material UI.

Those frameworks and design systems chosen by considering the following conditions: 1) Popularly used such as Bootstrap, Foundation, and Semantic UI; 2) There is a *kitchen sink page* [31] which contains usage example of frameworks or design system –where most of the dataset is taken from; 3) Easily found in search engine results; 4) Recommended in many online articles related to web design and development. The conditions considered with long term purpose that the system should be able to recognize most of GUI components implemented in the wild/real world websites. With that purposes, if the dataset is taken from widely used or known GUI framework or design system, than the system should have minimal problem in recognizing them in the wild.

Building Dataset from Sources

The images was taken by manually screenshotting GUI components examples from the kitchen sink page from respective CSS frameworks and design systems, or from each example page of GUI component from their websites. Those images then classified into specific categories; but with a condition where not all of those images of GUI components evenly taken –especially for button component. For example while from Bootstrap there are 29 image of buttons that could be extracted, there were none taken from Paper CSS because of it lacks of button state variations such as primary, danger, or disabled. Therefore, buttons data was only taken from Bootstrap, Foundation, and Semantic UI where those three frameworks have similar variations for button. Another consideration was if all button data is taken from mentioned sources, the amount will be outnumber all other component as button is the most common element in UI. Even with limited button data, the numbers of the button images already bigger than the other.

Crawling technique for automatic images mining was also considered. Previous works done the implementation by got the screenshot of web or apps, using object detection to segment the image and classify them. The technique, as explained before, was using top-down approach where they started from the whole page; meanwhile approach in this research was bottom-up where the dataset building started from the GUI components itself, not the whole page. For mining them directly from a web page using crawler, it needs novel implementation which can be considered as focus in another research.

Cleaning Dataset

For classification, at first there were defined eight class of GUI components which are: *button*, *textfield*, *textarea*, *checkbox*, *radio button*, *select,*

breadcrumbs, and *pagination.* As preliminary research, the categories are reduced to six with breadcrumbs and pagination being omitted to make the recognition effort easier. Also, the rest of six are common elements of UI which usually used in a digital form. Similar visual nature of those elements is considered such as textfield with textarea, and checkbox with radio button. Those similarities were tested for recognition in the experiment.

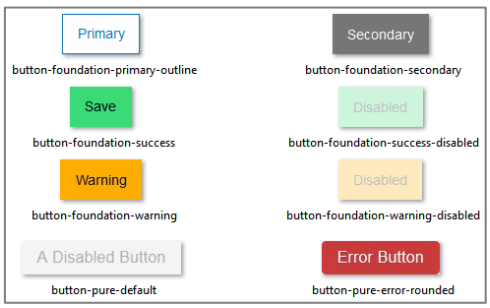

Figure 3. Example of button images in the dataset

Dataset for Experiment

The finished dataset contains images of 85 buttons, 70 textfields, 24 textarea, 49 checkboxes, 62 radio buttons, and 40 selects with example of the dataset can be seen at figure 3. Uneven numbers of images is caused by different characteristic of GUI components. For example buttons could have ten variations of states such as primary, secondary, success, warning, or disabled, while textarea usually only have three states: disabled, read only, and resizable. The ideal dataset should have even number across categories, and the number of data in a category should have sufficient amount to avoid under fitting problem in learning process. Images of GUI components are inserted into their respective classification folder, and they are inside a folder called gui-core-alpha where GUI CORE is the name of the dataset (from GUI COmponents REcognition), and alpha is the version of the dataset. The dataset is openly available online at https://github.com/agylardi/guicore-alpha.

Because of limited resources, the experiment still used the limited (alpha version) dataset to answer the main question in this research: whether deep learning technique can be used to recognize GUI components or not. The lack of data was tackled using data augmentation.

Data Input Preparations

For setting up the experiment, the dataset was splitted into two parts: 70% for training process, and 30% for validation. There were no test data because of limited numbers of the dataset itself. The dataset divided into training and validation folders, where each of them contains six categories of data. Figure 4 shows the structure of the dataset. For the deep learning experiment, it is implemented using Google Colab tools (Colab) which has cloud GPU that is available to use for deep learning process. The dataset uploaded to Google Drive (Drive) so it can be used through import and mount command in Colab.

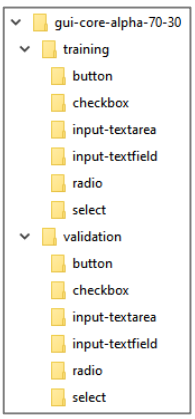

Figure 4. Structure of the dataset in the experiment

Colab is chosen because of limited hardware resources where the technique needs capable GPU for processing the calculation –usually from NVIDIA vendor which leads in deep learning infrastructure product. Colab allows access to NVIDIA Tesla K80 GPU by cloud connection but limited to 12 hours per day. For this preliminary research, that is more than sufficient. In implementation using Colab, jupyter notebook file which allows real-time python code processing was initiated.

For implementation, Keras framework and Tensorflow Library was chosen as it is suitable to be used in Colab environment. First process was importing data from Drive, and checked if the data loaded correctly by displaying some of the data using matplotlib library. Next step was setting basic parameter of the input with height and width 224 for resizing all of the input data to 224 x 224 px. The number was chosen to make input in chosen approaches has the same size.

Data Augmentation

Because of the lack of data, in this experiment the dataset was augmented using data augmentation process. This was done before the data processed by the models. In this research, training data was augmented with rescale 1./255 value for normalizing the input data. Training data also applied shear, zoom, and horizontal flip

transformation for the augmentation. For validation data, it was only applied with rescale for normalization purpose.

Setting CNN Models

After the data is ready, next is preparing the models. In this study, there were two approaches used: custom CNN, and transfer learning. For custom CNN it was a custom made architecture that defined before ran the model. The architecture can be seen in the following table 1, with the initial input was set at 224 x 224 x 3, and stride for every Conv layer is 3.

TABLE 1
CUSTOM CNN ARCHITECTURE

Layer Type	Output Shape
Conv + ReLu	224 x 224 x 32
Max Pool (stride 2)	112 x 112 x 32
Conv + ReLu	112 x 112 x 64
Max Pool (stride 2)	56 x 56 x 64
Conv + ReLu	56 x 56 x 128
Max Pool (stride 2)	28 x 28 x 128
Conv + ReLu	28 x 28 x 256
Max Pool (stride 2)	14 x 14 x 256
Dropout (rate 0.5)	14 x 14 x 256
Flatten	50176
Dense + ReLu	256
Dense (Softmax)	Classifier

For this approach, loss function was set to sparse categorical cross entropy, implemented in mentioned library with Adam optimizer, to make sure the classification output is exactly for each GUI component.

In second approach, transfer learning method was used. The method use existing trained model of CNN and improve upon it by trained it again using our dataset so it can do the proposed task. In this case, MobileNet [33] was used because its efficiency in deep learning implementation while still maintains acceptable accuracy performance. Architecture of MobileNet itself utilize what was called Depthwise Separable Convolution layer (Conv dw) which differ from normal convolution layer and helped in achieving efficiency. MobileNet architecture can be seen in table 2 with initial input, loss function and its optimizer were the same as custom CNN settings.

Both using batch valued at 32 implemented in Colab, and the experiment was conducted based on following conditions: 1) classification of two types of GUI; 2) classification of two similar type of GUI (textfield and textarea); 3) classification of six types of GUI.

TABLE 2
MOBILENET CNN ARCHITECTURE

Layer Type	Output Shape
Conv (stride 2)	112 x 112 x 32
Conv dw (stride 1)	112 x 112 x 32
Conv (stride 1)	112 x 112 x 64
Conv dw (stride 2)	56 x 56 x 64
Conv (stride 1)	56 x 56 x 128
Conv dw (stride 1)	56 x 56 x 128
Conv (stride 1)	56 x 56 x 128
Conv dw (stride 2)	28 x 28 x 128
Conv (stride 1)	28 x 28 x 256
Conv dw (stride 1)	28 x 28 x 256
Conv (stride 1)	28 x 28 x 256
Conv dw (stride 2)	14 x 14 x 256
Conv (stride 1)	14 x 14 x 512
5 x Conv dw (stride 1) Conv (stride 1)	14 x 14 x 512
Conv dw (stride 2)	7 x 7 x 512
Conv (stride 1)	7 x 7 x 1024
Conv dw 2 (stride 2)	7 x 7 x 1024
Conv (stride 1)	7 x 7 x 1024
Avg. Pool (7 x 7)	1 x 1 x 1024
Dense	1 x 1 x 1000
Dense (Softmax)	Classifier

4. Results and Analysis

Classification of Two Types of GUI

In this experiment scenario, only button and textfield categories were chosen, and epoch set at 50. Figure 5 at the right side shows the achieved accuracy in training and validation process with red line indicates training accuracy, and green line indicates validation accuracy; the left side shows the value from loss function.

It can be seen that the process experience under fitting because of insufficient dataset with accuracy of validation is below training; but with the accuracy of recognition (validation) could reach between 80 – 90%, the machine already capable discerning between button and textfield components.

For transfer learning approach, the result can be seen at Figure 6 below. It can be seen the result still under fitting but with boost on faster accuracy achievement. While custom CNN achieved accuracy of 90% above epoch 40, transfer

learning approach could reach it in 10 epochs. This shows how transfer learning could help greatly in recognition process of GUI components.

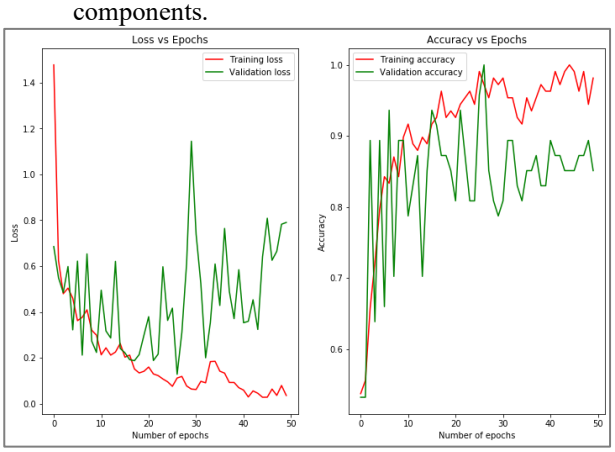

Figure 5. Result of custom CNN for two types of GUI

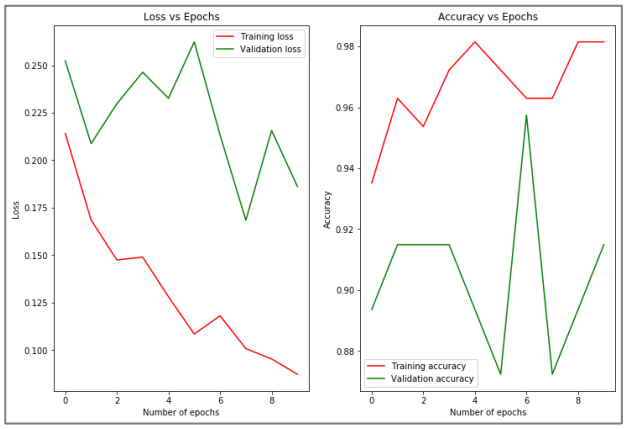

Figure 6. Result of MobileNet for two types of GUI

Classification of Two Similar Types of GUI

For this scenario, only textfield and textarea categories were used with the epoch set at 100. Figure 7 shows the result. While it seems did not experience under fitting, the accuracy of training and validation was both very fluctuated and could not stable until 100 epochs. It can be inferred that similar visual characteristic could leads to false classification of GUI components; because one component type can be identified as another type which showed in the fluctuation of accuracy.

By using transfer learning approach, the result was also similar as seen on Figure 8. Even though it was able to reach higher accuracy, but the fluctuation of accuracy is still happened between 80% and around 94% which was very wide. Still, transfer learning still considered better for the result.

Classification of Six Types of GUI

In this scenario, all of six mentioned categories of GUI components were included. With more categories and files to be processed, the result of custom CNN approach can be seen in Figure 8. It can be seen the validation accuracy suffered heavily and could not even reach stable 70% of accuracy while training accuracy still could reach 90% -an under fitting. It is expected as more categories need sufficient number of data, and current one is far from ideal. At the least, the classification of six types of GUI components is still shown possible to do using this technique.

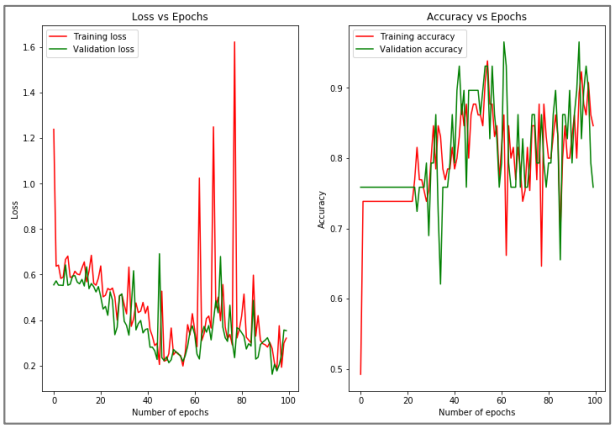

Figure 7. Result of custom CNN for two similar types

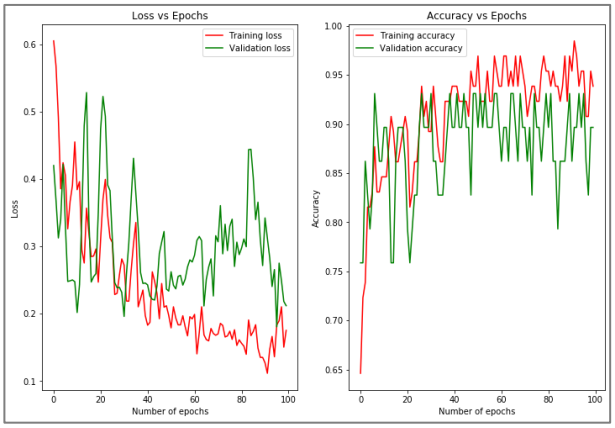

Figure 8. Result of MobileNet for two similar types

For transfer learning approach, recognition result of six types of GUI components can be seen in Figure 9. Using this approach gave better result with validation accuracy is higher (~70%) and more stable than custom CNN, but still suffered from under fitting. Once again, it is expected with the current dataset

5. Conclusion

This study proposes a foundation for building computer vision system using deep learning technique, with the final purpose for usabilty evaluation of UIs. This study contributes by 1) built GUI component images dataset, and 2) did deep learning experiment to validate whether it is suitable for basic recognition task of GUI components. From the results, it can be concluded that deep learning technique is suitable for visual recognition task of different types of GUI components.

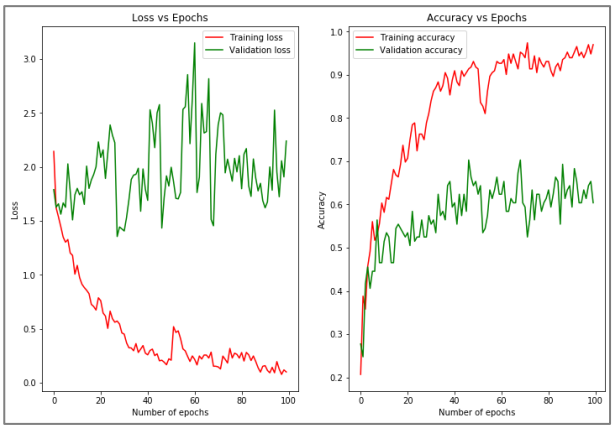

Figure 9. Result of custom CNN for six types of GUI

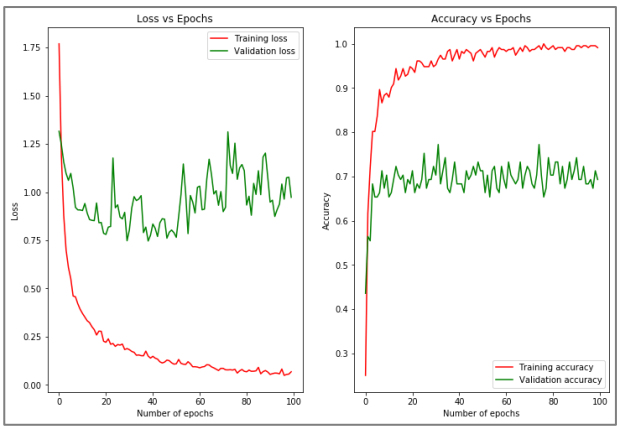

Figure 10. Result of MobileNet for six types of GUI

.

For future work it is suggested to build the dataset further so it is sufficient for the intended task, and to avoid under fitting in training process. For the method, it is preferable to use transfer learning approach in building computer vision application as it eliminates the trial and error of setting CNN architecture of the model –It is also could reach better result in shorter time. If possible, optimizations of parameters also done to explore which setting are best for the task. Other things that could be considered is availability of resources in doing deep learning technique as it requires expensive equipment if original architecture of CNN wanted to be freely explored.

Acknowledgement

Thank you for Universitas 17 Agustus 1945 Surabaya (Untag Surabaya) that funded this research in a form of internal research grant for the lecturers, and also for Institute of Research and Community Service (LPPM) of Untag Surabaya that was managing the whole grant process and being really helpful to grantees.

References

[1] J. Shariat and C. S. Saucier, *Tragic Design : The Impact of Bad Product Design and How to Fix It*, First. Sebastopol: O'Reilly Media, 2017.

[2] D. Norman and J. Nielsen, "Usability 101: Introduction to Usability," 2012. [Online]. Available: https://www.nngroup.com/articles/usability-101-introduction-to-usability/. [Accessed: 11-Dec-2019].

[3] B. Shneiderman, "The Eight Golden Rules of Interface Design," in *Shneiderman, B. and Plaisant, C., Designing the User Interface: Strategies for Effective Human-Computer Interaction: Fifth Edition*, 2010.

[4] E. Marcotte, *Responsive Web Design*, 2nd Edition. New York: A Book Apart, 2014.

[5] D. A. Norman, *The Design of Everyday Things*. USA: Basic Books, Inc., 2002.

[6] J. Nielsen, "Iterative User-Interface Design," *Computer (Long. Beach. Calif).*, vol. 26, no. 11, 1993, doi: doi.org/10.1109/2.241424.

[7] J. Cao, K. Zieba, and M. Ellis, *The Ultimate Guide to Prototyping*. Mountain View: UXPin Studio, 2015.

[8] S. Minhas, "User Experience Design Process," 2018. [Online]. Available: https://uxplanet.org/user-experience-design-process-d91df1a45916. [Accessed: 01-Dec-2019].

[9] C. Murphy, "A Comprehensive Guide To Wireframing And Prototyping," 2018. [Online]. Available: https://www.smashingmagazine.com/2018/03/guide-wireframing-prototyping/#top. [Accessed: 11-Dec-2019].

[10] G. Venturi and J. Troost, "Survey on the UCD integration in the industry," 2004.

[11] M. O. Riedl and R. St Amant, "Toward Automated Exploration of Interactive

Systems," 2002.

[12] K. Gibbs, T. Winograd, and N. Scott, "Lens: A System for Visual Interpretation of Graphical User Interfaces," 2002.

[13] T.-H. Chang, T. Yeh, and R. C. Miller, "GUI Testing Using Computer Vision," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2010.

[14] J. Koch and A. Oulasvirta, "Computational layout perception using Gestalt laws," in *Conference on Human Factors in Computing Systems - Proceedings*, 2016, vol. 07-12-May-2016, pp. 1423–1429, doi: 10.1145/2851581.2892537.

[15] M. Soegaard, "Laws of Proximity, Uniform Connectedness, and Continuation – Gestalt Principles (2)," *Interaction Design Foundation*, 2020. [Online]. Available: https://www.interaction-design.org/literature/article/laws-of-proximity-uniform-connectedness-and-continuation-gestalt-principles-2. [Accessed: 11-Dec-2019].

[16] A. Oulasvirta *et al.*, "Aalto Interface Metrics (AIM): A service and codebase for computational GUI evaluation," in *UIST 2018 Adjunct - Adjunct Publication of the 31st Annual ACM Symposium on User Interface Software and Technology*, 2018, pp. 16–19, doi: 10.1145/3266037.3266087.

[17] T. F. Liu, M. Craft, J. Situ, E. Yumer, R. Mech, and R. Kumar, "Learning design semantics for mobile apps," in *UIST 2018 - Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, 2018, pp. 569–579, doi: 10.1145/3242587.3242650.

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

[19] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553. Nature Publishing Group, pp. 436–444, 27-May-2015, doi: 10.1038/nature14539.

[20] U. Karn, "An Intuitive Explanation of Convolutional Neural Networks," *The Data Science Blog*, 2016. [Online]. Available: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/. [Accessed: 11-Dec-2019].

[21] R. A. Fernandez, J. A. Deja, and B. P. V. Samson, "Automating heuristic evaluation of websites using convolutional neural networks," in *Conference on Human Factors in Computing Systems - Proceedings*, 2018, pp. 9–12, doi: 10.1145/3205851.3205854.

[22] H. Lu, L. Wang, M. Ye, K. Yan, and Q. Jin, "DNN-based Image Classification for Software GUI Testing," in *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, 2018, pp. 1818–1823.

[23] S. Hassan, M. Arya, U. Bhardwaj, and S. Kole, "Extraction and Classification of User Interface Components from an Image," *Int. J. Pure Appl. Math.*, vol. 118, no. 24, 2018.

[24] T. T. Nguyen, P. M. Vu, H. V. Pham, and T. T. Nguyen, "Deep learning UI design patterns of mobile apps," in *Proceedings - International Conference on Software Engineering*, 2018, pp. 65–68, doi: 10.1145/3183399.3183422.

[25] T. A. Nguyen and C. Csallner, "Reverse Engineering Mobile Application User Interfaces With REMAUI," in *ASE '15: Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering*, 2015, pp. 248–259, doi: 10.1109/ASE.2015.32.

[26] T. Beltramelli, "pix2code: Generating Code from a Graphical User Interface Screenshot," *Comput. Researcy Repos.*, vol. abs/1705.07962, May 2017.

[27] K. Moran, C. Bernal-Cárdenas, M. Curcio, R. Bonett, and D. Poshyvanyk, "Machine Learning-Based Prototyping of Graphical User Interfaces for Mobile Apps," Feb. 2018.

[28] C. Chen, T. Su, G. Meng, Z. Xing, and Y. Liu, "From UI Design Image to GUI Skeleton: A Neural Machine Translator to Bootstrap Mobile GUI Implementation," in *Proceedings of ICSE '18: 40th International Conference on Software Engineering*, 2018, pp. 665–676, doi: 10.1145/3180155.3180240.

[29] A. Robinson, "Sketch2code: Generating a

website from a paper mockup," *Comput. Res. Repos.*, vol. abs/1905.13750, May 2019.

[30] B. Kim, S. Park, T. Won, J. Heo, and B. Kim, "Deep-learning based web UI automatic programming," in *Proceedings of the 2018 Research in Adaptive and Convergent Systems, RACS 2018*, 2018, pp. 64–65, doi: 10.1145/3264746.3264807.

[31] J. Win, "Libraries, Components, Boilerplates, Frameworks, and Kitchen Sink: How Do They Differ?," 2017. [Online]. Available: https://medium.com/@justinewin/libraries-components-boilerplates-frameworks-and-kitchen-sink-whats-their-difference-a0fa20128f11. [Accessed: 11-Dec-2019].

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks."

[33] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," May 2019.