IMPROVING PIX2CODE BASED BI-DIRECTIONAL LSTM

PROJECT REPORT

Submitted by

BHAGYA SREE
(PRN16CS020)

JINO JOHN
(PRN16CS027)

PARVATHI L
(PRN16CS038)

REEMA MIRIAM JOHN
(PRN16CS044)

to

the APJ Abdul Kalam Technological University

in partial fulfillment of the requirements for the award of the Degree

of

Bachelor of Technology

in

Computer Science and Engineering

Department of Computer Science and Engineering

College of Engineering Perumon

Kollam-691601

MAY 2020

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COLLEGE OF ENGINEERING PERUMON, KOLLAM

VISION

To emerge a department of global stature in the field of computer science education, research and development.

MISSION

- To equip graduates in the field of competent technical and analytical skills, innovative research capabilities and research potential.
- To instill graduates with integrity, discipline and ethics to work with commitment for the progress of the society.

DECLARATION

We undersigned hereby declare that the project report "**IMPROVING PIX2CODE BASED BI-DIRECTIONAL LSTM"**, submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under supervision of Mrs. Suja Vijayan. This submission represents our ideas in our own words and where ideas or words of others have been included,We have adequately and accurately cited and referenced the original sources. We also declare that We have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Kollam

28/05/2020

BHAGYA SREE

JINO JOHN

PARVATHY L

REEMA MIRIAM JOHN

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COLLEGE OF ENGINEERING PERUMON, KOLLAM

CERTIFICATE

This is to certify that the report entitled **"IMPROVING PIX2CODE BASED BIDIRECTIONAL LSTM"** submitted by **Bhagya Sree, Jino John, Parvathi L, and Reema Miriam John** to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering is a bonafide record of the project work carried out by her under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Project Guide:	**Project Coordinator(s):**		**HEAD OF THE DEPT:**
Mrs Suja Vijayan	Mrs Deepa K Daniel	Mrs Devi Dath	Mr Bejoy Abraham
Assistant Professor	Assistant Professor	Assistant Professor	Associate Professor & Head
Dept of IT	Dept of CSE	Dept of CSE	Dept of CSE/IT

ACKNOWLEDGEMENT

Firstly, we would like to thank almighty thus we were able to complete the major project within the given time.

We express our sincere gratitude to **Prof. Dr. Z A Zoya**, Principal of College of Engineering Perumon, for providing us with a well- equipped laboratory and all other facilities.

We also express our gratitude to **Mr. Bejoy Abraham**, Head of Computer Science and Information Technology Department, for providing all support and cooperation.

It is our atmost pleasure to convey our sincere thanks to **Mrs. Suja Vijayan**, our guide, for providing all valuable suggestions and support. We are also very much thankful to **Mrs. Devi Dath** and **Mrs. Deepa K Daniel**, our project coordinators for giving us moral support and cooperation. We also express our sincere gratitude to our tutor **Mrs. Sowmya K S** and co- tutor **Ms. Mili Mohan** for giving support and cooperation.

We also remember with regard our parents, for all the help and moral support that they provided for implementing our project. We express our gratitude to all other staff members and classmates for their help and motivation.

BHAGYA SREE

JINO JOHN

PARVATHY L

REEMA MIRIAM JOHN

ABSTRACT

It is common practice for developers of user-facing software to transform a mock-up of a graphical user interface (GUI) into code. Unfortunately, this practice is challenging and time-consuming. Pix2code is a framework based on deep learning to transform a graphical user interface screenshot created by the designer into computer code .In this paper, we present an approach that automates this process by enabling accurate prototyping of GUIs via three tasks: detection, classification, and assembly. First, logical components of a GUI are detected from a mock-up artifact using either computer vision techniques or mock-up metadata. Then, software repository mining, automated dynamic analysis, and deep convolutional neural networks (CNN) are utilized to accurately classify GUI- components into domain-specific types (e.g., toggle-button). The architecture is based on CNN and LSTM. LSTM has been broadly applied to natural language processing about language model, which is both general and effective at capturing long term dependencies. However, the standard LSTM predicting in time sequence ignores the contextual information of the future, but sometimes it is not enough just to look at the previous word. Computer code is a relative spatial relationship and not only needs to recognize token but also fully understands the structure of all sequences. In order to solve the problem, the pix2code model is optimized by Bidirectional LSTM, which allows the output layer to get complete past and future context information for each point in the input sequence. The model is able to generate code targeting two different platforms (i.e. Android and web-based technologies) from a single input image with better accuracy.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The process of implementing client-side software based on a Graphical User Interface (GUI) mock up created by a designer is the responsibility of developers. Implementing GUI code is, however, time-consuming and prevent developers from dedicating the majority of their time implementing the actual features and logic of the software they are building. Moreover, the computer languages used to implement such GUIs are specific to each target platform; thus resulting in tedious and repetitive work when the software being built is expected to run on multiple platforms using native technologies. In this paper, we describe a system that can automatically generate platform-specific computer code given a GUI screenshot as input. In industrial settings with larger teams, this process is typically carried out by dedicated designers who hold domain specific expertise in crafting attractive, intuitive GUIs using image- editing software such as Photoshop or Sketch. These teams are often responsible for expressing a coherent design language across the many facets of a company's digital presence, including websites, software applications and digital marketing materials. Many fast- moving startups and fledgling companies attempting to create software prototypes in order to demonstrate ideas and secure investor support would also greatly benefit from rapid application prototyping. Rather than spending scarce time and resources on iteratively designing and coding user interfaces, an accurate automated approach would likely be preferred. This would allow smaller companies to put more focus on features and value and less on translating designs into workable application code. Given the frustrations that front- end developers and designers face with constructing accurate GUIs, there is a clear need for automated support. Unfortunately, automating the

prototyping process for GUIs is a difficult task. At the core of this difficulty is the need to bridge a broad abstraction gap that necessitates reasoning accurate user interface code from either pixel based, graphical representations of GUIs or digital design sketches. Typically, this abstraction gap is bridged by a developer's domain knowledge. For example, a developer is capable of recognizing discrete objects in a mock-up that should be instantiated as components on the screen, categorizing them into proper categories based on their intended functionalities, and arranging them in a suitable hierarchical structure such that they display properly on a range of screen sizes.

CHAPTER 2

LITERATURE SURVEY

There are four papers related to our system is selected for literature review. They are described below.

2.1 Pix2code:GENERATING SCREENSHOT FROM GRAPHICAL USER INTERFACE[1]

This paper implement an encoder/decoder model that they trained on information from GUI-metadata and screenshots to translate target screenshots first into a domain specific language (DSL) and then into GUI code. In this paper, it shows that Deep Learning techniques can be leveraged to automatically generate code given a graphical user interface screenshot as input. Our model is able to generate code targeting three different platforms (i.e. iOS, Android and web-based technologies) from a single input image.

The task of generating code given a GUI screenshot as input can be compared to the task of generating English textual descriptions given a scene photography as input. Thus divide the problem into three sub-problems. First, a computer vision problem of understanding the given scene (i.e. in this case, the GUI screenshot image) and inferring the objects present, their identities, positions, and poses (i.e. buttons, labels, element containers). Second, a language modeling problem of understanding text (i.e. in this case, computer code) and generating

syntactically and semantically correct samples. Finally, the last challenge is to use the solutions to both previous sub-problems by exploiting the latent variables inferred from scene

understanding to generate corresponding textual descriptions (i.e. computer code rather than English text) of the objects represented by these variables.During training, the GUI screenshot picture is encoded by a CNN-based vision model; the sequence of one-hot encoded tokens corresponding to DSL code is encoded by a language model consisting of a stack of LSTM layers. The two resulting encoded vectors are then concatenated and fed into a second stack of LSTM layers acting as a decoder. Finally, a softmax layer is used to sample one token at a time; the output size of the softmax layer corresponding to the DSL vocabulary size. Given an image and a sequence of tokens, the model is differentiable and can thus be optimized end-to-end through gradient descent to predict the next token in the sequence. The input state (i.e. a sequence of tokens) is updated at each prediction to contain the last predicted token. During sampling, the generated DSL code is compiled to the desired target language using traditional compiler design techniques.This approach exhibits several shortcomings that call into question the real-world applicability of the approach: (i) the approach was only validated on a small set of synthetically generated applications, and no large-scale user interface mining was performed;(ii) the approach requires a DSL which will need to be maintained and updated over time, adding to the complexity and effort required to utilize the approach in practice.

2.2 REVERSE ENGINEERING MOBILE APPLICATION USER INTERFACES WITH REMAUI[2]

This system REMAUI, the first technique for inferring mobile application user interface code from screenshots or conceptual drawings. To evaluate REMAUI, it implemented a prototype tool that generates the UI portion of Android applications. This paper therefore identifies and addresses three problems in mobile application development. In reverse engineering, it address the problem of inferring the user interface code of a mobile application from screenshots. In forward engineering, we address the gap between scanned pencil-on paper UI sketches and code as well as the gap between pixel based UI sketches and code. REMAUI automatically infers the user interface portion of the source code of a mobile application from screenshots or conceptual drawings of the user interface. On a given input bitmap REMAUI identifies user interface elements such as images, text, containers, and lists, via computer vision and optical character recognition (OCR) techniques.

REMAUI uses a combination of OpticalCharacter Recognition (OCR), CV, and mobile specific heuristics to detect components and generate a static app.

Main methods are

- Optical character recognition

  Locate and extract candidate words and lines with optical character recognition.

- Computer vision

Locate and extract candidate UI elements as a hierarchy of nested bounding boxes using computer vision.

- Merging

  Merge the results to improve recognition quality.

- Identifying List

  Identify repeated items and summarize them as collection.

- Export

  Export the constructed UI as a mobile application for a given platform. Finally compile and execute then we will get the corresponding code.

REMAUI Advantages

- Reduce the gap between graphic artist contextual drawings and working UI.
- Less expensive
- Average overall runtime on standard desktop computer was 9 seconds.

REMAUI has key limitations**:**

- It does not support the classification of detected components into their native component types and instead uses a binary classification of either text or images, limiting the real-world applicability of the approach.
- It is unclear if the GUI hierarchies generated by REMAUI are realistic or useful from developer's point of view, as the GUI-hierarchies of the approach were notevaluate.

2.3 DETECTION OF GUI ELMENETS ON SKETCH IMAGES USING OBJECT DETECTOR BASED ON DEEP NEURAL NETWORKS[4]

Graphical user interface(GUI) is very important to interact with software users.Here GUI elements are converted to code or to describe formally its structure by help of domain knowledge or machine learning based algorithms. It uses object detection based on deep neural networks that find GUI elements by integration of localization and classification. After the successfully detection of GUI components, the objects are described as the hierarchical structure and transform those to appropriate codes by synthetic or machine learning algorithms.

In the proposed system, GUI objects are translated from sketch image to hierarchical object descriptions with object detection algorithm and next generate the GUI descriptions depended on the platforms through several managers. The system contains three stages. They are:

- Data Preparation

  There are 50 mimicked sketch images including about 600 GUI elements from the screenshot images. To recognize the graphic objects, mark GUI elements manually and save the bounding box and labels.

- Object Detection

  To find the graphic objects in sketch images , YOLO real time object detection technique is employed. YOLO applied a single neural network to the full image. YOLO divides the image into regions and predicts bounding boxes and probabilities for each region.

- Code Generation

  After finding graphic objects, bounding box and class information is transformed to hierarchical semantic structure. Then the meta information is translated to given platform dependent GUI codes.

Advantages

The previous work only depend on semantic domain knowledge, or they pursue end–to-end algorithm that gets the screenshot image and generates the given platform codes. But in the proposed system, we at first translate the GUI objects from sketch images to hierarchical structure with object detection algorithm and next generate the GUI descriptions depended on the platforms through several managers.

Disadvantages

Main source of error is incorrect localization. YOLO imposes strong spatial constraints on bounding box predictions which limits the no of nearby objects that our model can predict.

2.4 CROWD SOURCED USER INTERFACE THAT COMES TO LIFE AS YOU SKETCH THEM[3]

This system, called Apparition,uses real-time crowd sourcing infrastructure allowing fluid prototyping of interfaces containing interactive elements, complex animations, and intelligent feedback.Crowd workers and sketch recognition algorithms translate the input into user interface elements, add animations providing wizard-of-0z functionality.Designers begin by sketching a low-fidelity version of their interface using either a pen and tablet or a mouse.Automatic gesture recognition identifies user strokes when possible using specially trained $1 Recognizer.If the recognizer cannot confidently classify and convert the sketch into the UI element Apparition asks workers to produce a higher fidelity rendering in real-time by

using simple visual prototyping tools provided to them.Visual elements replace sketches within 8 seconds on average, and interactive behaviours function in 3 seconds.

Apparition contains:

- Collaborative canvas where the user(s) and workers can draw.
- Drawing tools and icons to quicken workers' creation of UI elements.
- A search function to help workers find relevant icons.
- A to-do list that shows what has been drawn by the user but not yet converted to UI elements. Workers can "accept" tasks to signal what they are currently working on.
- "In-progress" markers for workers to show where they are currently working to avoid conflicts.

Advantages

Apparition, make even hard-to-automate functions work immediately without the need for one-off control implementations. Apparition will allow designers to create prototypes fast enough to iterate within single design sessions, improving the feedback they receive.The prototypes created by Apparition are over 90% accurate to the user's intent.

Disadvantages

Workers focused on their self defined tasks and often missed looking at the to-do list on the side of the screen at a critical time. This results in blocking actions causing work to be undone. Requiring that a worker first tag and route a task introduced unacceptable latency.

CHAPTER 3

MATERIALS AND METHODS

3.1 SYSTEM SPECIFICATION

3.1.1 Software Specification

Operating System : Ubuntu

Technology : Python

IDETool : Jupyter Notebook

3.1.2 Hardware Specification

System : Intel i3, 3GHz

Hard Disk : 100 GB

Monitor : 15 VGA Colour

Mouse : HP

Ram : 4GB

3.2 FEASIBILITY STUDY

A feasibility study is not warranted for systems in which economic purification is obvious, technical risk is low, few legal problems are expected no reasonable alternative exists. An estimate is made of whether the identified user needs may be satisfied using current software and hardware technologies. The feasibility study should be relatively cheap and quick. The result should inform the decision ofwhether to go ahead with a more detailed analysis.

Feasibility study may be documented as a separated report to higher officials of the top level management and can be included as an appendix to the system specification. Feasibility and risk analysis is related on many ways. If there is more project risk then the feasibility of producing the quality software is reduced.The main objective of feasibility study is to test the technical, social and economic feasibility of developing a system. This is done before developing a system. This is done by investigating the existing system in the area under investigation and generating ideas about the new system.

- Technical Feasibility

  Technical feasibility centers around the existing computer system to what extent it will support the existing system. If it needs additional hardware, which involves an additional consideration to accommodate technical enhancements then this project is judged not feasible. In Technical feasibility the current level of technology can support the proposed system is checked. Considering all the advantages of the proposed system and also the profit as a serious constraint, we recommended to use the proposed system. This project completetly an research based predict system, The main technology used here are Ubuntu, Python, Jupyter. Each of the technologies are freely available and the technical skills are managable. Time limitations of the product development and the ease of implementing using these technologies are synchronised.

- Economic Feasibility

  A cost evaluation is weighed against the ultimate income or benefit divided from the developed system or product. Economic justification is generally the

"bottom- line" consideration that includes cost benefit analysis, long term corporate income strategies, impact on other profit centers or products, cost of resources needed for development and potential market growth. A projection of the amount of funding or startup capital needed, what sources of capital a business can and will use, and what is the return on investment. Organizational feasibility: A definition of the corporate and legal structure of the business. This may include information about the founders, their professional background and the skills they possess necessary to get the company off the ground and keep it operational.

- Behavioral Feasibility

  There is no resistance for implementing the new system as it offers greater user friendliness. The authority is also convinced of one benefit resulting from the implementation of the system and so it is operationally feasible, so by all means the proposed system is to the firm.

- Operational Feasibility

  Proposed project would be beneficial only if it can be turned into information system that will meet the organization's operational requirements. One of the main problems faced during development of a new system is getting acceptance from user. Even if a system is technically and economically feasible but the users of the system are resistant to use it then there is no use. In this stage the following issues are considered.

  1) Is the proposed system is user friendly?

2) Is there sufficient support for the project from the management and users?

3) Will the proposed system cause harm?

4) Will it produce poorer result in any area?

5) Will loss of control result in any area?

The proposed system is so effective, user friendly and functionally reliable that the users in the company will find that the new system reduce their work. The result produced is accurate and optimized. The proposed system causes no harm to the organization. The proposed system will have good control on all parts of the organization and it will take care of current activities.

3.3 DATASET

We have used 1750 web images and 1750 android images along with their corresponding GUI code files. Training dataset is stored in the file training_set and testing dataset is stored in eval_set. Along with this there are 1750 sketch images and their corresponding gui codes.

Figure 3.1: Plot of web and android images

An example of web image and android image are given below.

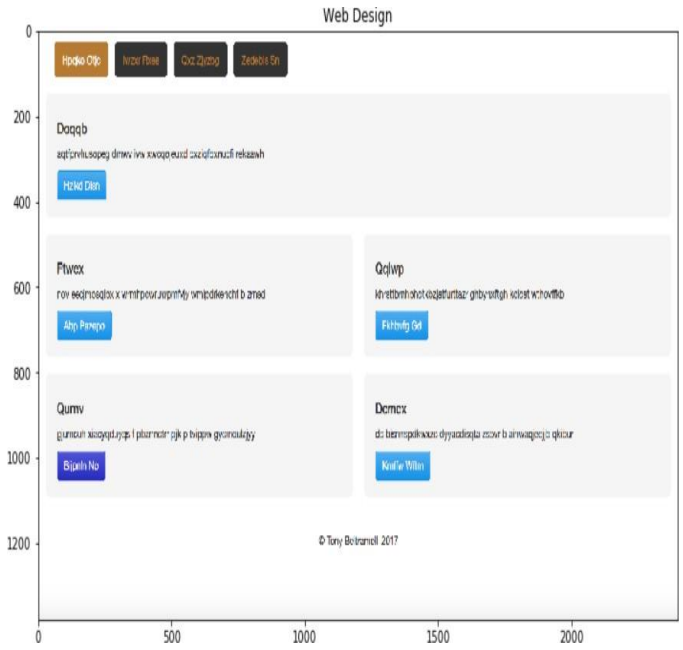


Figure 3.2: Example of a web image

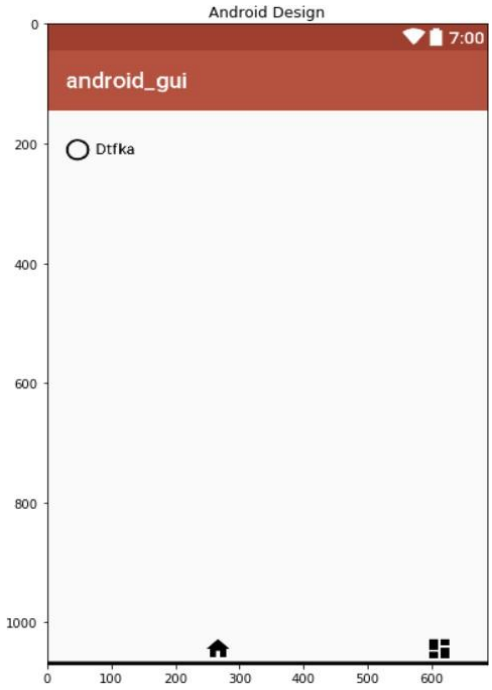


Figure 3.3:Example of android

3.4 LANGUAGE USED

3.4.1 Python

Python is an object-oriented, high-level programming language with integrated dynamic semantics primarily for web and app development. It is extremely attractive in the field of Rapid Application Development because it offers dynamic typing and dynamic binding options. Python is relatively simple, so it's easy to learn since it requires a unique syntax that focuses on readability. Developers can read and translate Python code much easier than other languages. In turn, this reduces the cost of program maintenance and development because it allows teams to

Work collaboratively without significant language and experience barriers. Additionally, Python supports the use of modules and packages, which means that programs can bedesigned in a modular style and code can be reused across a variety of projects. Once you've developed a module or package you need, it can be scaled for use in other projects, and it's easy toimport or export these modules. One of the most promising benefits of Python is that both the standard library and the interpreter are available free of charge, in both binary and source form. Thereis no exclusivity either, as Python and all the necessary tools are available on all major platforms. Therefore, it is an enticing option for developers who don't want to worry about paying high development costs.If this description of Python over your head, don't worry. You'll understand it soon enough. What you need to take away from this section is that Python is a programming language used to develop software on the web and in app form, including mobile. It's relatively easy to learn, and the necessary tools are available to all free of charge. That makes Python accessible to almost anyone. If you have the time to learn, you can create some amazing things with the language. Python is a general-purpose programming language, which is another way to say that it can be used for nearly everything. Most importantly, it is an interpreted language, which means that the written code is not actually translated to a computer-readable format at run time. Whereas, most programming languages do this conversion before the program is even run. This type of language is also referred to as a "scripting language" because it was initially meant to be used for trivial projects. The concept of a "scripting language" has changed considerably since its inception, because Python is now used to write large, commercial style applications, instead of just banal ones. This reliance on Python has grown even more so as the

internet gained popularity. A large majority of web applications and platforms rely on Python, including Google's search engine, YouTube, and the web-oriented transaction system of the NewYork Stock Exchange (NYSE). You know the language must be pretty serious when it's powering a stock exchange system. In fact, NASA actually uses Python when they are programming their equipment and space machinery.

FEATURES OF PYTHON

- A simple language which is easier to learn - Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C]. Python makes programming fun and allows you to focus on the solution rather than syntax.

- Free and open-source - You can freely use and distribute Python, even for commercial use. Not only can you use and distribute softwares written in it, you can even make changes to the Python's source code. Python has a large community constantly improving it in each iteration.

- Portability - Can move Python programs from one platform to another, and run it without any changes. It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.

- Extensible and Embeddable - Suppose an application requires high performance. an easily combine pieces of C/C++ or other languages with Python code. This will give your application high performance as well as scripting capabilities which other languages may not provide out of box.

- A high-level, interpreted language - Unlike C/C++, you don’t have to worry about daunting tasks like memory management, garbage collection and so on. Likewise, when you run Python code, it automatically converts your code to the language your com- puter understands. You don’t need to worry about any lower-level operations.

- Large standard libraries to solve common tasks - Python has a number of standard libraries which makes life of a programmer much easier since you don’t have to write all the code yourself. For example:If Need to connect MySQL database on a Web server?

  MySQLdb library using import MySQLdb . Standard libraries in Python are well tested and used by hundreds of people. So you can be sure that it won’t break your application.

3.5 TOOLS USED

3.5.1 Anaconda

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Anaconda distribution comes with more than 1,400 packages as well as the Conda package and virtual environment manager, called Anaconda Navigator, so it eliminates the need to learn to install each library independently.The open source packages can be individually installed from the Anaconda repository with the conda install command or using the pip install command that is installed

with Anaconda. Pip packagesprovide many of the features of conda packages and in most cases they can work together.Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud,PyPI or other repositories.The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with conda.

3.5.2 Anaconda Navigator

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux.Anaconda Cloud is a package management service by Anaconda where you can find, access, store and share public and private notebooks, environments, and conda and PyPI packages. Cloud hosts useful Python packages, notebooks and environments for a wide variety of applications. You do not need to log in or to have a Cloud account, to search for public packages, download and install them.

3.5.3 Jupyter

Jupyter is language agnostic and its name is a reference to core programming languages supported by Jupyter which are Julia, Python and R, and it supports execution environment (aka kernels) in several dozen of languages among which are Julia, R, Haskell, Ruby, and of course Python (via the IPython kernel).

The Jupyter Notebook combines three components:

1. The notebook web application: An interactive web application for writing and running code interactively and authoring notebook documents.

2. Kernels: Separate processes started by the notebook web application that runs users' code in a given language and returns output back to the notebook web application. The kernel also handles things like computations for interactive widgets, tab completion and introspection.

3. Notebook documents: Self-contained documents that contain a representation of all content visible in the notebook web application, including inputs and outputs of the computations, narrative text, equations, images, and rich media representations of objects. Each notebook document has its own kernel.

The notebook web application enables users to:

1. Edit code in the browser, with automatic syntax highlighting, indentation, and tab completion/ introspection.

2. Run code from the browser, with the results of computations attached to the code which generated them.

3. See the results of computations with rich media representations, such as HTML, LaTeX, PNG, SVG, PDF, etc.

4. Create and use interactive JavaScript widgets, which bind interactive user interface controls and visualizations to reactive kernel side computations.

5. Author narrative text using the Markdown markup language.

6. Include mathematical equations using LaTeX syntax in Markdown, which are rendered in-browser by MathJax.

3.6 PACKAGES USED

3.6.1 NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object.
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random

3.6.2 Pandas

Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables.In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and oper- ations for manipulating numerical tables and time series.

Library Features

- DataFrame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures

- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, fancy indexing, and subsetting of large data sets.
- Data structure column insertion and deletion.
- Group by engine allowing split-apply-combine operations on data sets. 8.Data set merging and joining.
- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
- Time series-functionality: Date range generation[4] and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging.
- Provides data filtration.

3.6.3 MatPlotLib

Matplotlib is a magic function in IPython. matplotlib inline sets the backend of matplotlib to the 'inline' backend: With this backend, the output of plotting commands is displayed inline within frontends like the Jupyter notebook, directly below the code cell that produced it. matplotlib, which is used for forming high-quality plots.

3.6.4 SeaBorn

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.Seaborn is a library for making statistical graphics in Python. It is built on top of matplotlib and closely integrated with pandas data structures.

Functionality of SeaBorn

1. A dataset-oriented API for examining relationships between multiple variables

2. Specialized support for using categorical variables to show observations or aggregate statistics .

3. Options for visualizing univariate or bivariate distributions and for comparing them between subsets of data

4. Automatic estimation and plotting of linear regression models for different kinds dependent variables.

5. Convenient views onto the overall structure of complex datasets

6. High-level abstractions for structuring multi-plot grids that let you easily build complex visualizations

7. Concise control over matplotlib figure styling with several built-in themes 8.Tools for choosing color palettes that faithfully reveal patterns in your data

3.6.5 Tensorflow

TensorFlow is the second machine learning framework that Google created and used to design, build, and train deep learning models. You can use the TensorFlow library do to numerical computations, which in itself doesn't seem all too special, but these computations are done with data flow graphs. In these graphs, nodes represent mathematmathematical operations, while the edges represent the data, which usually are multidimensional data arrays or tensors, that are communicated between these edges.

3.6.6 Keras

Keras is an Open Source Neural Network library written in Python that runs on top of Theano or Tensorflow. Keras High-Level API handles the way we make models, defining layers, orset up multiple input-output models. In this level, Keras also compiles our model with loss and optimizer functions, training process with fit function. Keras doesn't handle Low-Level API such as making the computational graph, making tensors or other variables because it has been handled by the "backend" engine.

Functions of Keras

- Large Community Support

  There are lots of AI communities that use Keras for their Deep Learning framework. Many of them publish their codes as well tutorial to the general public

- Have multiple Backends

  You can choose Tensorflow, CNTK, and Theano as your backend with Keras.You can choose a different backend for different projects depending on your

  needs. Each backend has its own unique advantage.

- Cross-Platform and Easy Model Deployment

  With a variety of supported devices and platforms, you can deploy Keras on any device like iOS with CoreML, Android with Tensorflow Android, Web browser with .js support, Cloud engine, Raspberry Pi

CHAPTER 4

DESIGN AND METHODOLOGY

4.1 SYSTEM DESIGN

Design is the first step in the development phase for any engineering product or system. It is the process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient detail to permits its physical realization. Design is a creative process and is the key to a good and effective system. Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm that is used. The word design is defined as the process of applying various techniques and principles for the purpose of defining a process or a system in sufficient detail to permit its physical realization. Design starts with the system requirement specification and converts it to a physical reality during the development.

Usually, software design is conducted in two steps – preliminary design and detail design. Preliminary design is concerned with the transformation of requirements into data and software architecture. Detail design focuses on refinement to the architectural representation that lead to detail algorithm data structure and representation of software.

The system design transforms a logical representation of what a given system is required to be in the physical specification. In system design high-end decisions are taken regarding the basic system architecture, platforms and tools to be used. Important design factors such as reliability, response time, throughput of the system maintainability, expandability etc should also be taken into account during design. System design refers to the description of a new system based on the information that is collected during the analysis phase and the process by whichit is developed.

System design builds on the information gathered during system analysis. It is the creative process of inventing and developing new inputs, database and output to meet the system objectives. It provides the understanding the procedural details necessary for implementing the system recommended in the feasibility study. Here the emphasis is on translating the performance requirement into design specification.

Design goes through logical and physical stages of development. Logical design reviews the present physical systems, prepares input and output specification, makes edit security and controlspecification, detailed implementation plan and prepares a logical design walk through. The physical design maps out the details of the physical system, plans the system implementation, devices a test and implementation plan and specifies any new hardware and software. System design can be considered as the important point in the system development cycle.

4.2 METHODOLOGY

The task of generating code given a GUI screenshot as input can be compared to the task of generating English textual descriptions given a scene photography as input. We can thus divide our problem into three sub-problems. First, a computer vision problem of understanding the given scene (i.e. in this case, the GUI screenshot image) and inferring the objects present, their identities, positions, and poses (i.e. buttons, labels, element containers). Second, a language modeling problem of understanding text (i.e. in this case, computer code) and generating syntactically and semantically correct samples. Finally, the last challenge is to use the solutions to both previous sub-problems by exploiting the latent variables inferred from scene understanding to generate corresponding textual descriptions (i.e. computer code rather than English text) of the objects represented by these variables.

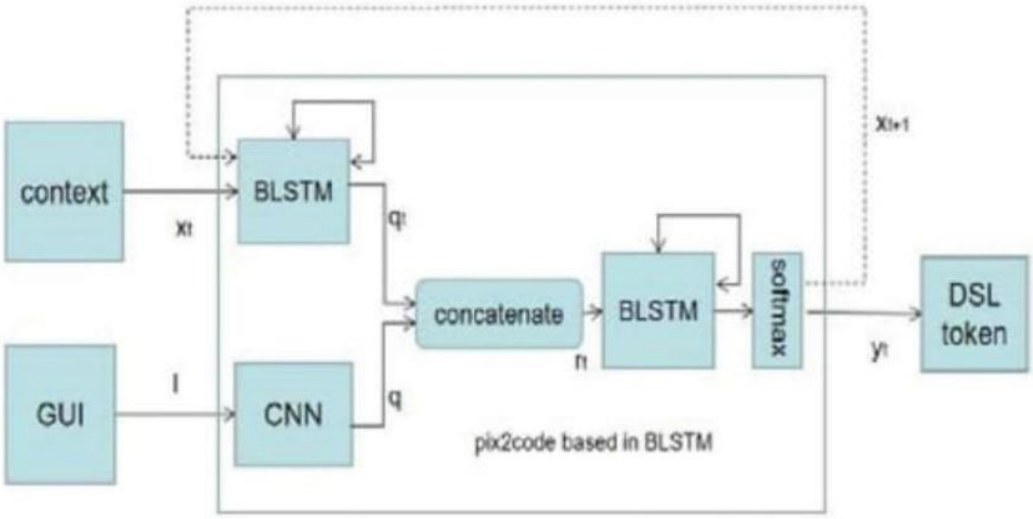


Figure 4.1: BLSTM Based Pix2code Model

4.2.1 Vision Model

CNNs are currently the method of choice to solve a wide range of vision problems thanks to their topology allowing them to learn rich latent representations from the images they are trained on. We used a CNN to perform unsupervised feature learning by mapping an input image to a learned fixed-length vector; thus acting as an encoder. The input images are initially re-sized to 256 × 256 pixels (the aspect ratio is not preserved) and the pixel values are normalized before to be fed in the CNN. No further pre-processing is performed. To encode each input image to a fixed-size output vector, we exclusively used small 3 × 3 receptive fields which are convolved with stride 1. These operations are applied twice before to down-sample with max-pooling. The width of the first convolutional layer is 32, followed by a layer of width 64, and finally width 128. Two fully connected layers of size 1024 applying the rectified linear unit activation complete the vision model.

4.2.2 Language Model

We designed a simple DSL to describe GUIs as illustrated in Figure 5. In this, we are only interested in the GUI layout, the different graphical components, and their relationships; thus the actual textual value of the labels is ignored by our DSL. Additionally to reducing the size of the search space, the DSL simplicity also reduces the size of the vocabulary (i.e. the total number of tokens supported by the DSL). As a result, our language model can perform token-level language modeling with a discrete input from using one-hot encoded vectors.

It is important to capture long-term dependencies to be able to close a block that has been opened. LSTM is both general and effective at capturing long-term temporal dependencies. The current word can be predicted based on the previous word context information of the text when processing the sequence model, but for some tasks only based on the previous word is not very accurate, because the preceding and following words in a sentence are not independent. If the model has access to the context of the previous word context and the words behind it, it is very helpful for language modeling. The basic idea of the BLSTM is to use two LSTMs forward and backward for each training sequence, and both of them are connected to a fully connected layer. This structure allows the output layer to get complete past and future context information for each point in the input sequence.

Figure 4.2 : An example of web gui with DSL code

4.2.3 Dropout In Decoder

Dropout is a very powerful tool that is effective in improving the performance of deep neural networks. The output dimension of the decoder layer is 512, and the output dimension of the language model is 128, so there may be overfitting. Dropout is useful for solving overfitting problems while training complex deep networks. A 25% dropout is set between the two layers of BLSTM at the decoder layer. It can be observed that the performance of the model has been obviously improved, the correct rate of the evaluation set has improvedprominently.

4.2.4 Decoder (BLSTM)

The latent features for both context and images are concatenated and feed to a decoder.The decoder contains a stack of two BLSTM layers with output dimension at 512 for each time step. Then it is feed into a fully connected layer to compute probabilities for each vocabulary using softmax. We select the output DSL token with the highest probability. For example, if our vocabulary size is just 5, the model will make a prediction of (0.05, 0.1, 0.05, 0.3, 0.5) to represent the probability for each word in the vocabulary.

4.3 TRAINING

The architecture performs supervised learning during training. Its input is GUI I and a sequence xt. The CNN based visual model extracting feature from the image I generates a vectorial representation p .The input token xt is encoded by a BLSTM-based language model into an intermediary representation qt. The vision-encoded vector p and the language-encoded vector qt are concatenated into a single feature vector rt which is then fed into a second BLSTM-based model (decoder) decoding the representations learned by both the vision model and the language model. The decoder thus learns to model the relationship between objects present in the input GUI image and the associated tokens present in the DSL code.

This architecture based on BLSTM can be expressed mathematically in Equations.

$$p = CNN(I)$$

$$qt = BLSTM(xt)$$

$$rt = (q,pt)$$

$$yt = softmax(BLSTM(rt))$$

$$xt+1 = yt$$

When the language model and the decoder process a sequence, it is possible to obtain previous word context information and the subsequent word context of the currently input for modeling and predicting.For training the initial dataset with 1750 web and 1750 android images along with their corresponding gui files are split into two.1500 images are selected for training and the remaining 250 images for tesing. The model is trained with mini batches of 64 images.

4.4 TESTING

Testing is the activity where the errors remaining from all the previous phases must be detected. Testing performs a very critical role for ensuring quality. During testing, the software to be tested is executed with a set of test cases, and the behaviour of the system for the test

cases is evaluated to determine if the system is performing as expected. The common view of testing held by users is that it is performed to prove that there are no errors in a program. However, as indicated earlier, that is virtually impossible, since analyst cannot prove that software is free and clear of errors. The tester, who may be the program fail. A successful test, then, is one that finds an error. Analysts know that an effective testing program does not guarantee system reliability. Therefore, reliability must be designed in to the system.

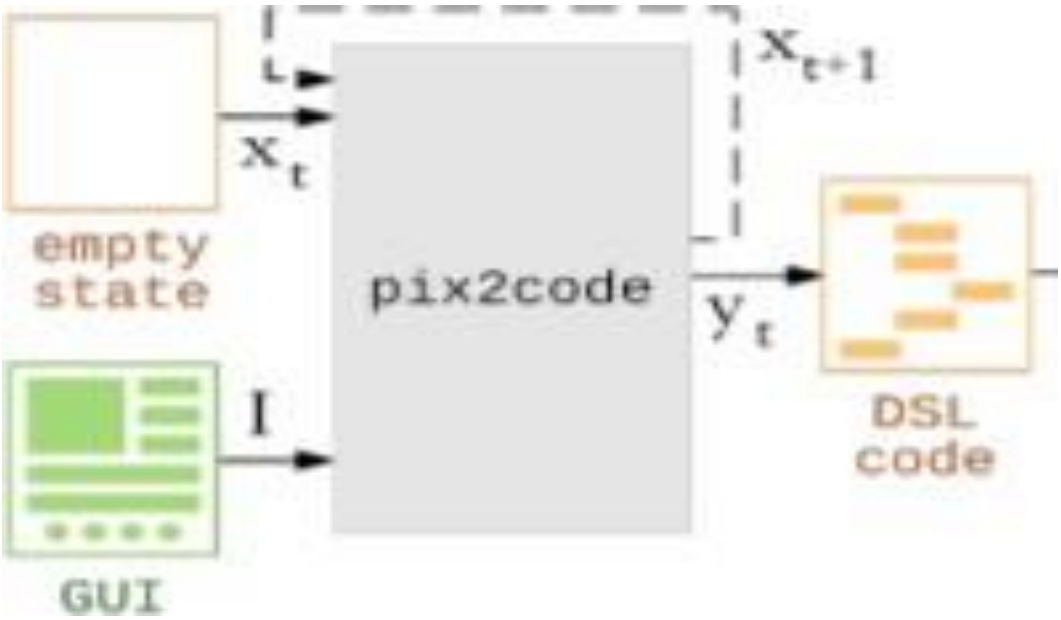


Figure 4.3: Testing architecture

For testing 250 images from the actual dataset with 1750 images are selected. These were the images that were not given to the model for training. To generate DSL code, we feed the GUI image I and a sequence X of T = 19 tokens where tokens xt . . . xT −1 are initially set empty and the last token of the sequence xT is set to the special < ST ART > token. The predicted token yt is then used to update the next sequence of input tokens. That is, xt . . . xT −1 are set to xt+1 . . . xT (xt is thus discarded), with xT set to yt. The process is repeated until the token < END > is generated by the model. The generated DSL code can then be compiled with traditional compilation methods to the desired target language.

CHAPTER 5

RESULT AND DISCUSSION

Results show that using BLSTM enhances the robustness of pix2code and increases accuracy.Our model takes as input a Web or sketch image and generates the DSL code which can be compiled to target code using any traditional compilation techniques.

Figure 5.1:Web image sample input

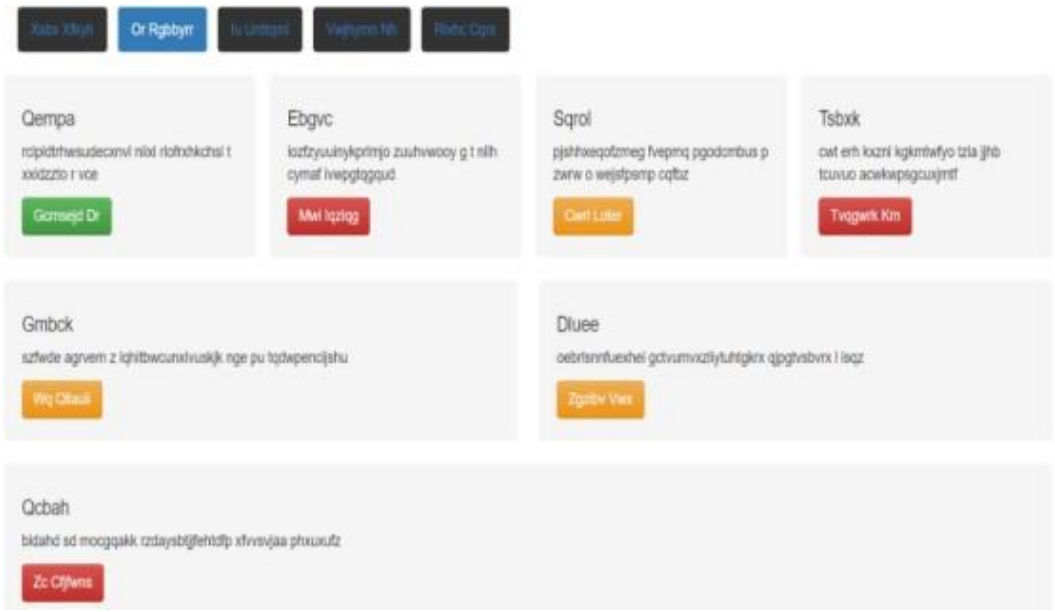


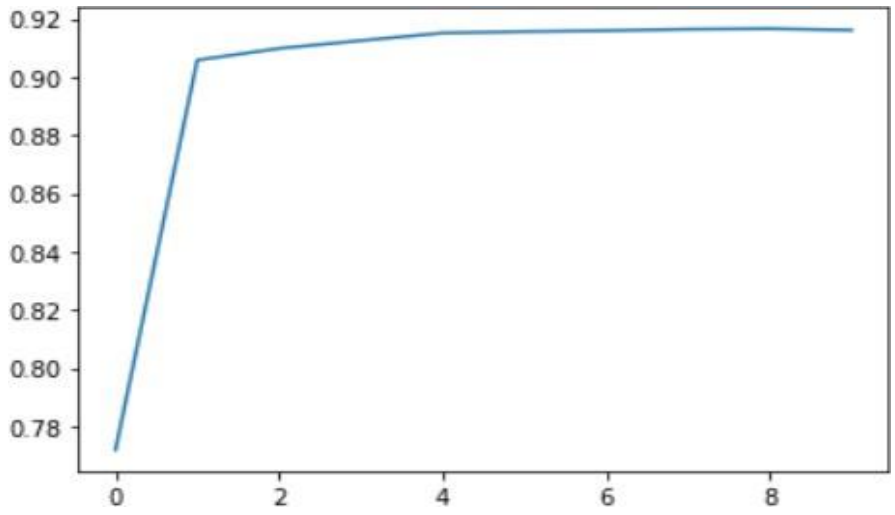
Figure 5.2: Predicted GUI image

```
header {
btn-inactive, btn-active, btn-inactive, btn-inactive, btn-inactive
}
row {
quadruple {
small-title, text, btn-green
}
quadruple {
small-title, text, btn-red
}
quadruple {
small-title, text, btn-orange
}
quadruple {
small-title, text, btn-red
}
}
row {
double {
small-title, text, btn-orange
}
double {
small-title, text, btn-orange
}
}
row {
single {
small-title, text, btn-red
}
}
```

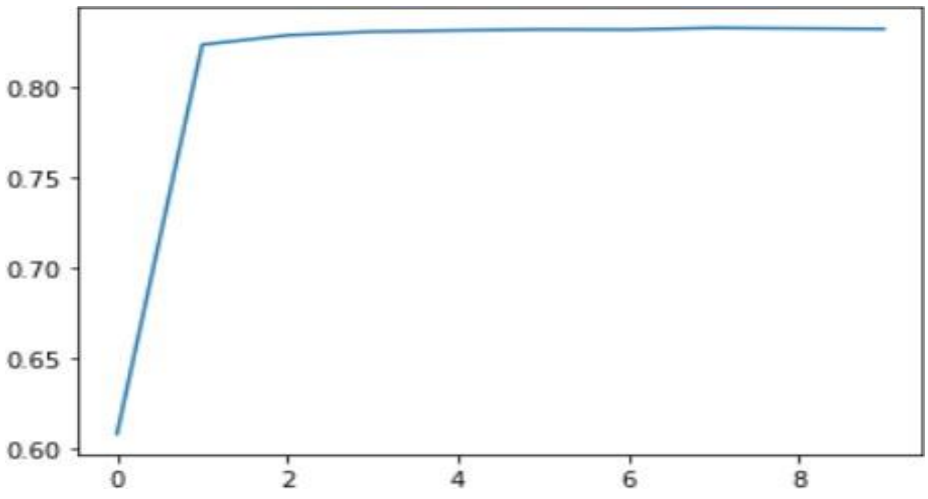
Figure 5.3:Predicted GUI code

The output layer of our model has the same number of cells as the vocabulary size; thus generating a probability distribution of the candidate tokens at each time step allowing the use of a softmax layer to perform multi-class classification.Accuracy is one of the metrices to measure the performance of the model. From 77% on the first epoch, the accuracy increased to 91% on the tenth epoch for Web. For android, accuracy increased from 60% to 83%.

$$\text{Accuracy} = \frac{\text{No of correct predictions}}{\text{Total no of predictions}}$$



(i)



(ii)

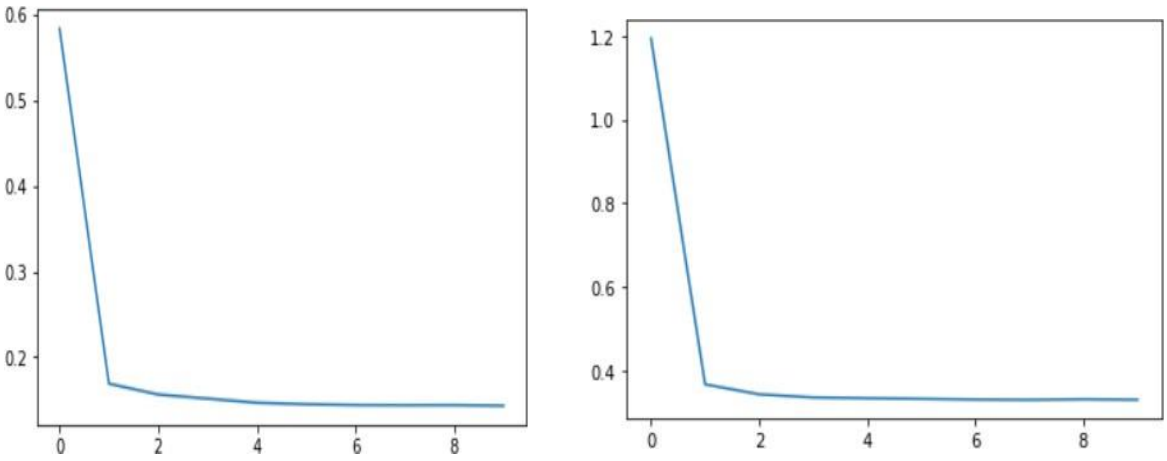
Figure 5.4:Accuracy plots (i)Web and (ii)Android

Loss is the difference between the predicted value by the model and true value. The loss function used here is categorical cross-entropy and defined as:

$$L = -\sum_{t=1}^{T} x_{t+1} \log (y_t)$$

With x_{t+1} the expected token, and y_t the predicted token. The model is optimized end-to-end hence the loss L is minimized with regard to all the parameters including all layers in the CNN-based vision model and all layers in both BLSTM-based models. Training with the RMSProp algorithm gave the best results with a learning rate set to 1r =0.001 and clipvalue=1.0 to cope with numerical instability .

To prevent overfitting, a dropout regularization set to 25% is applied to the vision model after each max-pooling operation and at 30% after each fully-connected layer. A 25% dropout layer between the two layers of BLSTM in the decoder model is added.



(i) (ii)

Figure 5.5: Loss plots (i) web and (ii) android

The model also works for sketch mockups and predicts DSL code and output. When a sample sketch image is given to the model as shown in figure 13, it generates output as in figure 14 amd figure 15.

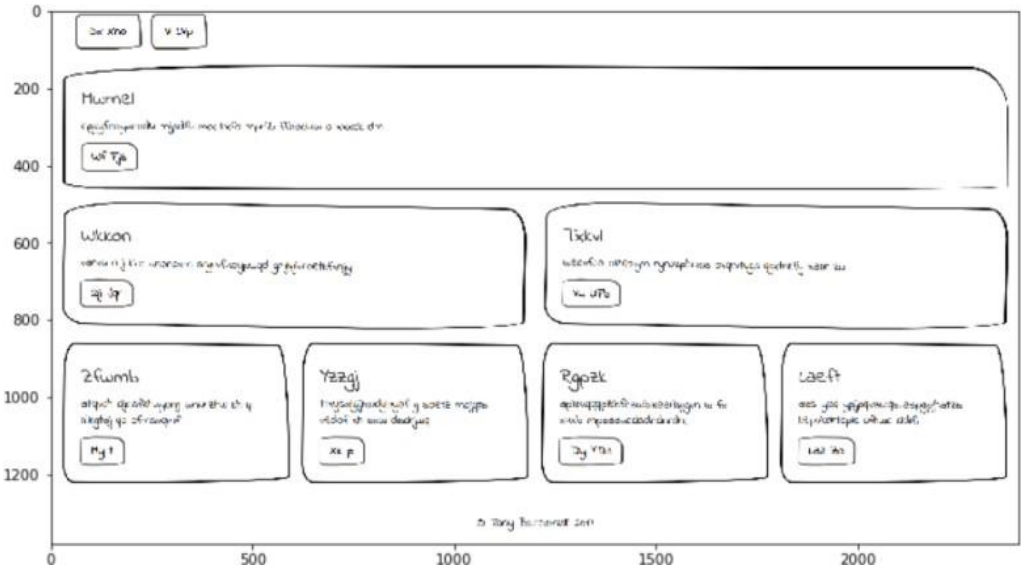


Figure 5.6: Sketch image sample input

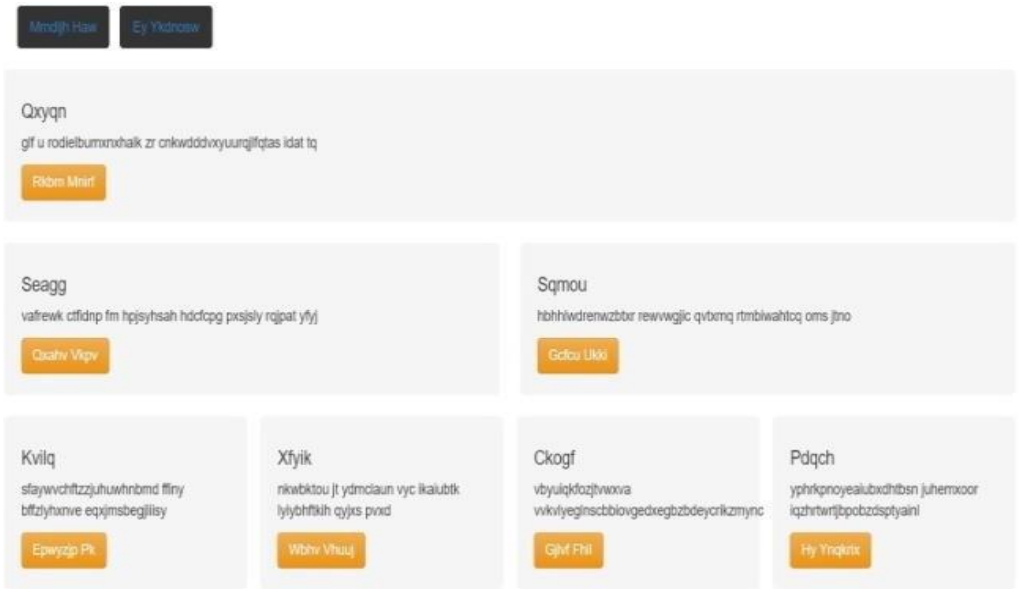


Figure 5.7: Predicted GUI image

```
<START>
header {
 btn-inactive , btn-inactive }
row {
single {
small-title , text , btn-orange } }
row {
double {
small-title , text , btn-orange }
double {
small-title , text , btn-orange } }
row {
quadruple {
small-title , text , btn-orange }
quadruple {
small-title , text , btn-orange }
quadruple {
small-title , text , btn-orange }
quadruple {
small-title , text , btn-orange } }
<END>
```

Figure 5.8: Predicted GUI code

The actual textual value of the labels is ignored and that both our data synthesis algorithm and our DSL compiler assigns randomly generated text to the labels. Despite occasional problems to select the right color or the right style for specific GUI elements and some difficulties modelling GUIs consisting of long lists of graphical components, our model is generally able to learn the GUI layout in a satisfying manner and can preserve the hierarchical structure of the graphical elements.

CHAPTER 6

CONCLUSION

In this paper, we presented a novel method to generate computer code given a single GUI screenshot as input. While our work demonstrates the potential of such a system to automate GUI programming, we only scratched the surface of what is possible. Our model consists of relatively few parameters and was trained on a relatively small dataset. The quality of the generated code could be drastically improved by training a bigger model on significantly more data for an extended number of epochs.This is a novel method to generate computer code given a single GUI screenshot as input. The quality of the generated code would be drastically improved by training a bigger model on significantly more data for an extended number of epcohs. A comprehensive evaluation of this method is capable of 1)accurately detecting and classifying GUI components in a mockup artfact, 2) generating hierarchies that are similar to those that a developer would create ,3) generating apps that are visually similar to mockup artifacts, and 4) positively impacting industrial workflows. this explores CNN architectures aimed at object detection to better support the detection task. Using one-hot encoding as we did does not provide any useful information about the relationships between the tokens since the method simply assigns an arbitrary vector representation to each token. Therefore, pre-training the language model to learn vectorial representations of the tokens would allow the relationships between tokens in the DSL to be inferred and as a result alleviate semantical error in the generated code. Furthermore, one-hot encoding does not scale to very big vocabulary and thus restrict the number of symbols that the DSL can support. Considering a large number of websites already available online and the fact that new websites are created every day, the web could theoretically supply an unlimited amount of training data. We extrapolate that Deep Learning used in this manner could eventually end the need for manually-programmed GUIs.

REFERENCES

[1] Beltramelli T. “pix2code: Generating Code from a Graphical User Interface Screenshot.”Proc. ACM SIGCHI Symp. Eng. Interact. Comput. Syst,June 2018.

[2] T. A. Nguyen and C. Csallner, ‘‘Reverse engineering mobile application user interfaces with REMAUI,’’ in Proc. ASE, Singapore, 2016.

[3] Walter S. Lasecki, Juho Kim, Nick Rafter, Onkur Sen, Jeffrey P. Bigham, and Michael S. Bernstein. 2015. Apparition: Crowd sourced User Interfaces That Come to Life As You Sketch Them. In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. ACM, New York, NY, USA.

[4] Y.-S. Yun, J. Jung, S. Eun, S.-S. So, and J. Heo. Detection of gui elements on sketch images using object detector based on deep neural networks, International Conference on Green and Human Information Technology, Springer, 2018.

[5] Yingtao, Xie & Lin, Tao & Xu, Hongyan. (2019). User Interface Code Retrieval: A Novel Visual-Representation-Aware Approach. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.

[6] Steven P Reiss.2014. Seeking the user interface. In Proceedings of the 29th ACM/IEEE international conference on Automated software engineering. ACM.

[7] Sethi, Nandini & Kumar, Abhishek & Swami, Rohit. (2019). Automated web development: theme detection and code generation using Mix-NLP.

[8]] S. Suleri, V. P. Sermuga Pandian, S. Shishkovets, M. Jarke, Eve: A sketch-based software prototyping workbench, intended Abstracts of the ACM Int. Conf. on Human Factors in Computing Systems, ACM, New York, NY, USA, 2019

[9] M. Balog, A. L. Gaunt, M. Brockschmidt, S. Nowozin, and D. Tarlow. “ Deepcoder: Learning to write programs.

[10] K. P. Moran, C. Bernal-Cárdenas, M. Curcio, R. Bonett, D. Poshy-vanyk, "Machine learning-based prototyping of graphical user interfaces for mobile apps", IEEE Transactions on Software Engineering.

[11] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

[12]] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In Proceedings of the IEEE conference on computer vision and pattern recognition.

[13] Jeffrey Nichols and Andrew Faulring, “Automatic interface generation and future user interface tools,” ACM CHI 2005 Workshop on The Future of User Interface Design Tools, (2005).

[14] Vinícius C. V. B. Segura, Simone D. J. Barbosa, and Fabiana Pedreira Simões. 2012. UISKEI: A Sketch-based Prototyping Tool for Defining and Evaluating User Interface Behavior. (2012).

[15] X. Yin, X. Yin, K. Huang, H. Hao, "Robust Text Detection in Natural Scene Images", IEEE Transactions on Pattern Analysis and Machine Intelligence, 2014.

[16] K. Simonyan, A. Zisserman, "Very deep convolutional networks for large-scale image recognition", ICLR, 2015.

[17] Donahue, J, Hendricks, L. A, Rohrbach, M, Venugopalan S, Guadarrama, S., & Saenko, K., et al.”Long-term Recurrent Convolutional Networks for Visual Recognition and Description.” Proc IEEE Comput Soc Conf Comput Vision Pattern Recognition.

[18] Gers F A, Schmidhuber J, Cummins F. “Learning to forget: continual prediction with LSTM.” IEE Conf Publ 1999.

[19] A. Karpathy and L. Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In Proceedings of the IEEE Conference on Computer Vision and PatteRecognition.

[20] A. Mart´ınez, H. Estrada, J. Sanchez, and O. Pastor, “From early requirements to user interface prototyping: A methodological approach,” in Proc. 17th IEEE International Conference on Automated Software Engineering (ASE). IEEE, Sep. 2002.