Image to CODE

AUTOMATIC HTML CODE GENERATION FROM IMAGES USING DEEP LEARNING BASED MODEL

Contents

- Introduction
- Problem Statement and Objective
- Dataset
- Methodology
- Implementation
- Results
- Future Scope
- References

Introduction

Let’s just take an overview of how any website is made:

- **I**t all starts by **designing the UI and UX of the website** and creating mockups for individual website pages by using graphics or simply handmade.
- Frontend developer will then use these images to **code the frontend portion using HTML**, which doesn't require much expertise.

This is also a tedious, tiresome and cumbersome process and this gave us the idea to **automate the code generation process** so that developer or engineer can invest his or her time in more creative tasks.

Problem Statement & Objective

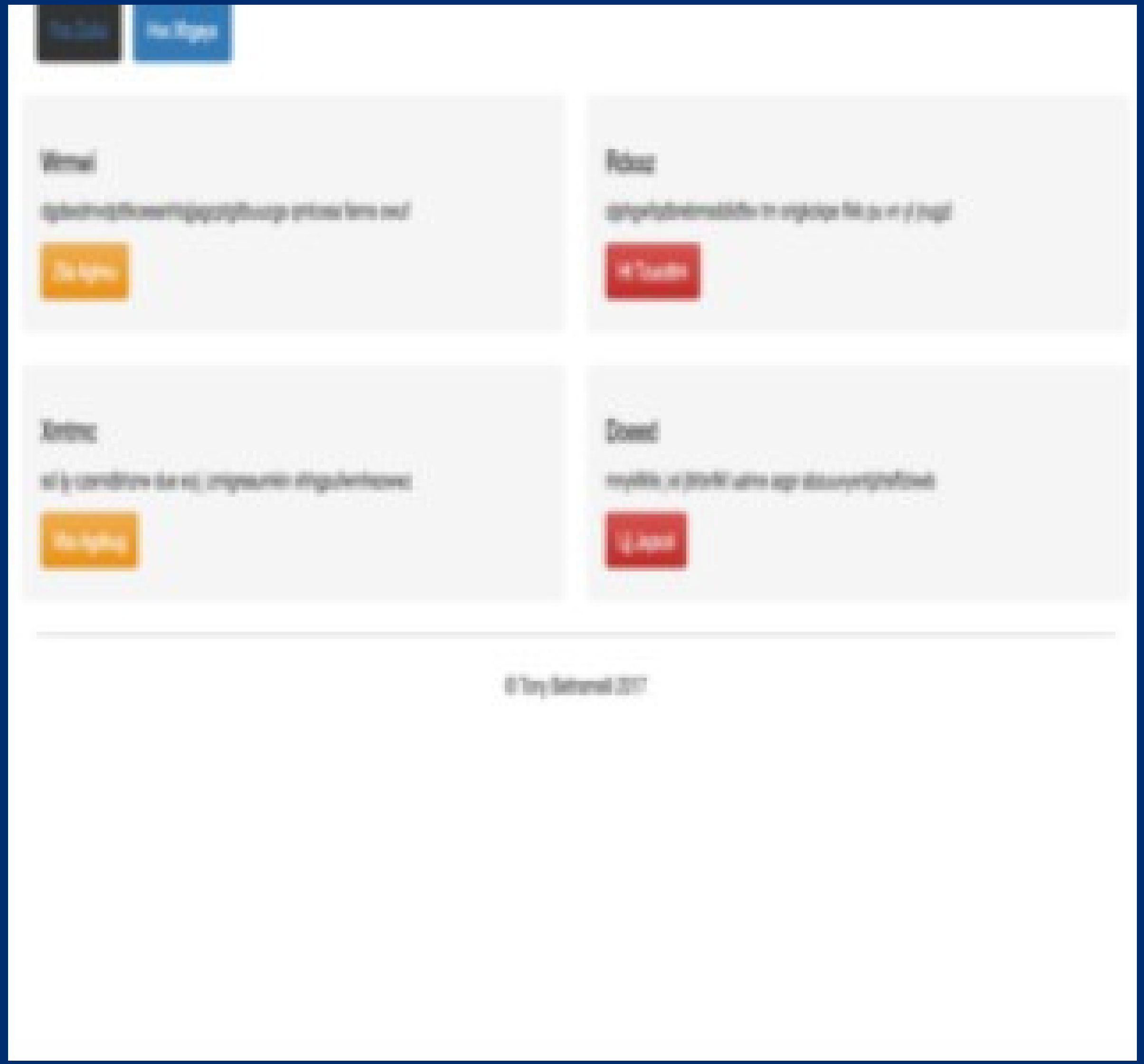
We aim at automating the tedious and cumbersome process of front end generation.

01 Hiring frontend engineers increases the cost for a company which can be reduced or automated by using some deep learning techniques.

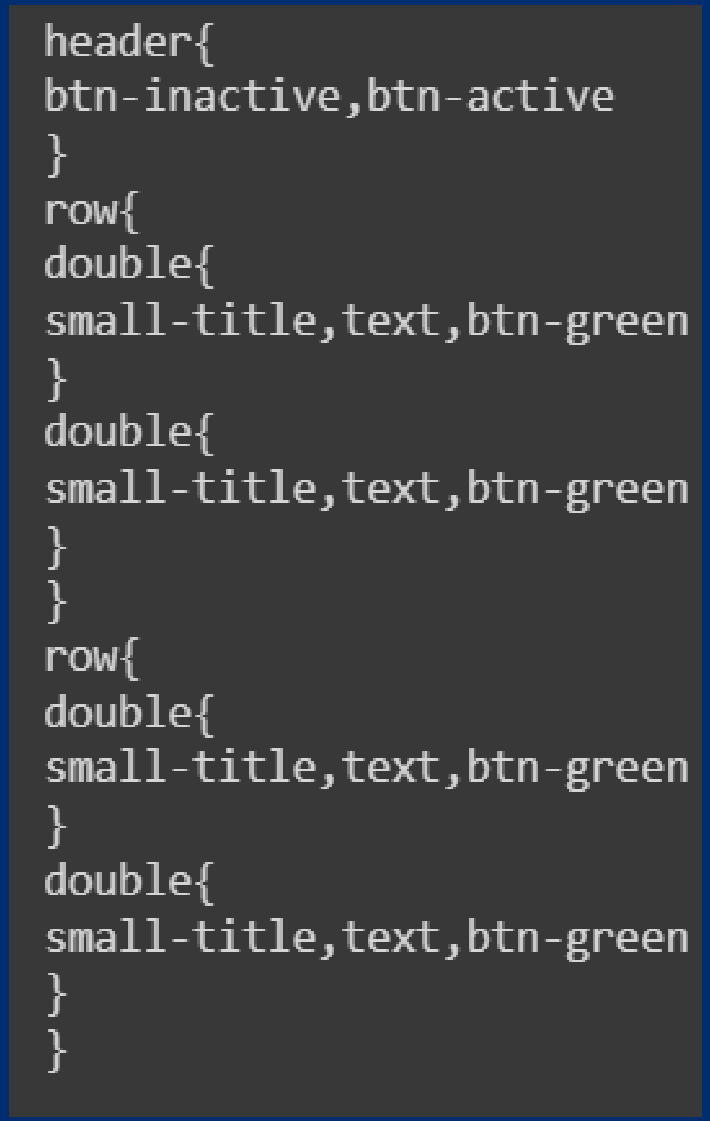
02 To automate the frontend development for websites in a secure manner.

In short

We will use a **CNN and RNN encoder-decoder model** to address challenges such as recognizing dependencies and creating sequential text.



INPUT



```
header{
btn-inactive,btn-active
}
row{
double{
small-title,text,btn-green
}
double{
small-title,text,btn-green
}
}
row{
double{
small-title,text,btn-green
}
double{
small-title,text,btn-green
}
}
```

OUTPUT

Dataset: Pix2code

Dataset Link: https://www.kaggle.com/code/himasha0421/pix2code/input

The Pix2code dataset is a collection of image-HTML pairs that are used for training and evaluating ML models that aim to automatically generate code from GUI designs.

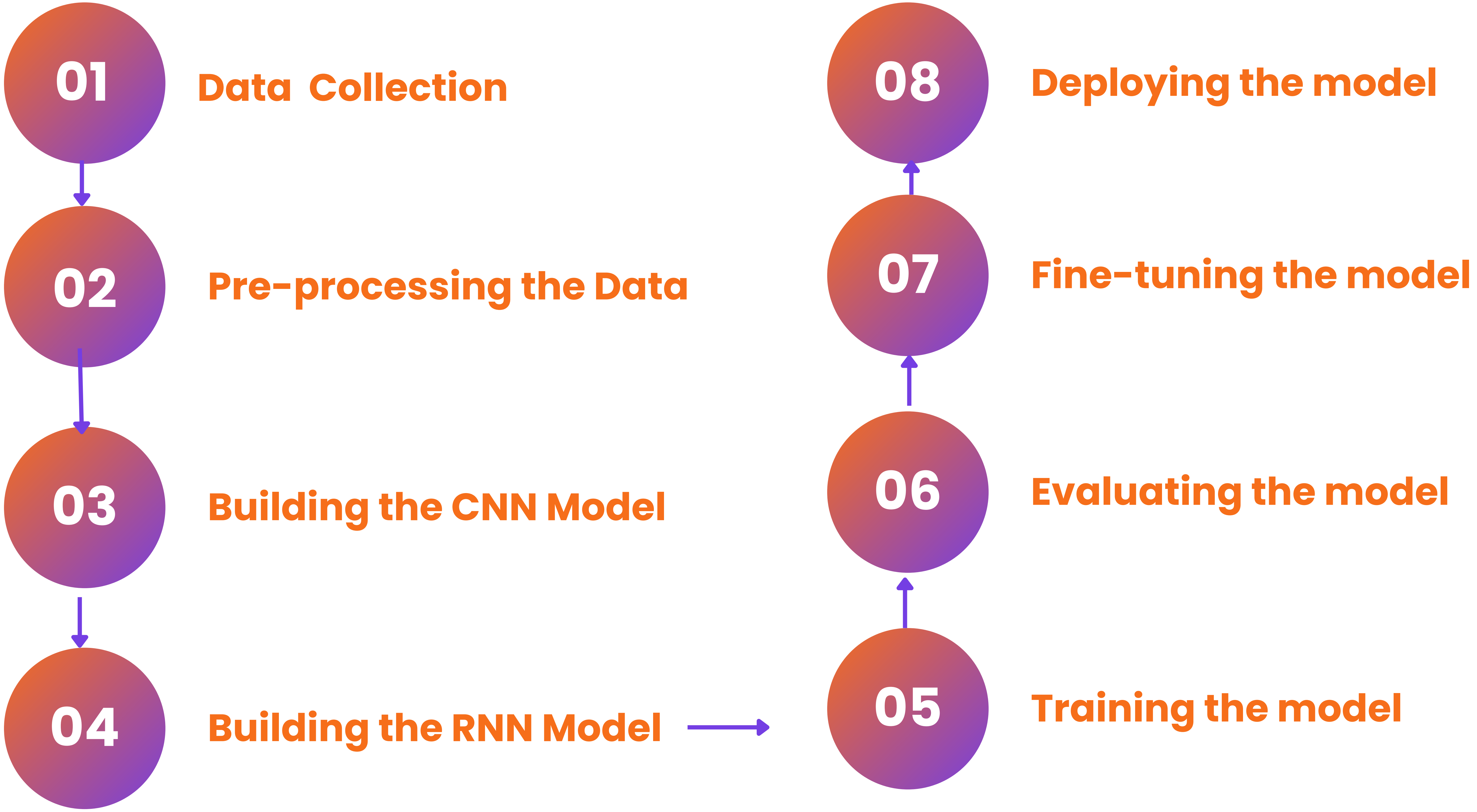
The dataset contains **15,000** images of GUI designs, each of which is paired with the corresponding HTML code that describes the layout and functionality of the design.

The images are taken from a wide variety of sources.

The dataset contained three portions: Android, IOS and web, we will currently be working on the web segment.

The dataset is split into three parts: a training set (**6000**) , a validation set (**500**), and a test set (**500**).

Methodology



01 Data Collection

02 Pre-processing the Data

03 Building the CNN Model

04 Building the RNN Model

05 Training the model

06 Evaluating the model

07 Fine-tuning the model

08 Deploying the model

01

Data Collection

Collect a dataset of UI design images and their corresponding HTML code.

02

Pre-processing the Data

Pre-process the images and HTML code to make them suitable for training.

03

Building the CNN Model

CNN model to extract features from the **input image**. The output will be a **feature vector** that will be input to RNN Model.

04

Building the RNN Model

RNN model that generates **HTML Code** for the **input feature vector**s.

05

Training the model

Train the CNN-RNN model after pre processing the dataset. Pre processing involves cleaning the HTML code using **one-hot-encoding.**

06

Evaluating the model

Evaluate the model's performance on a separate validation dataset .

07

Fine-tuning the model

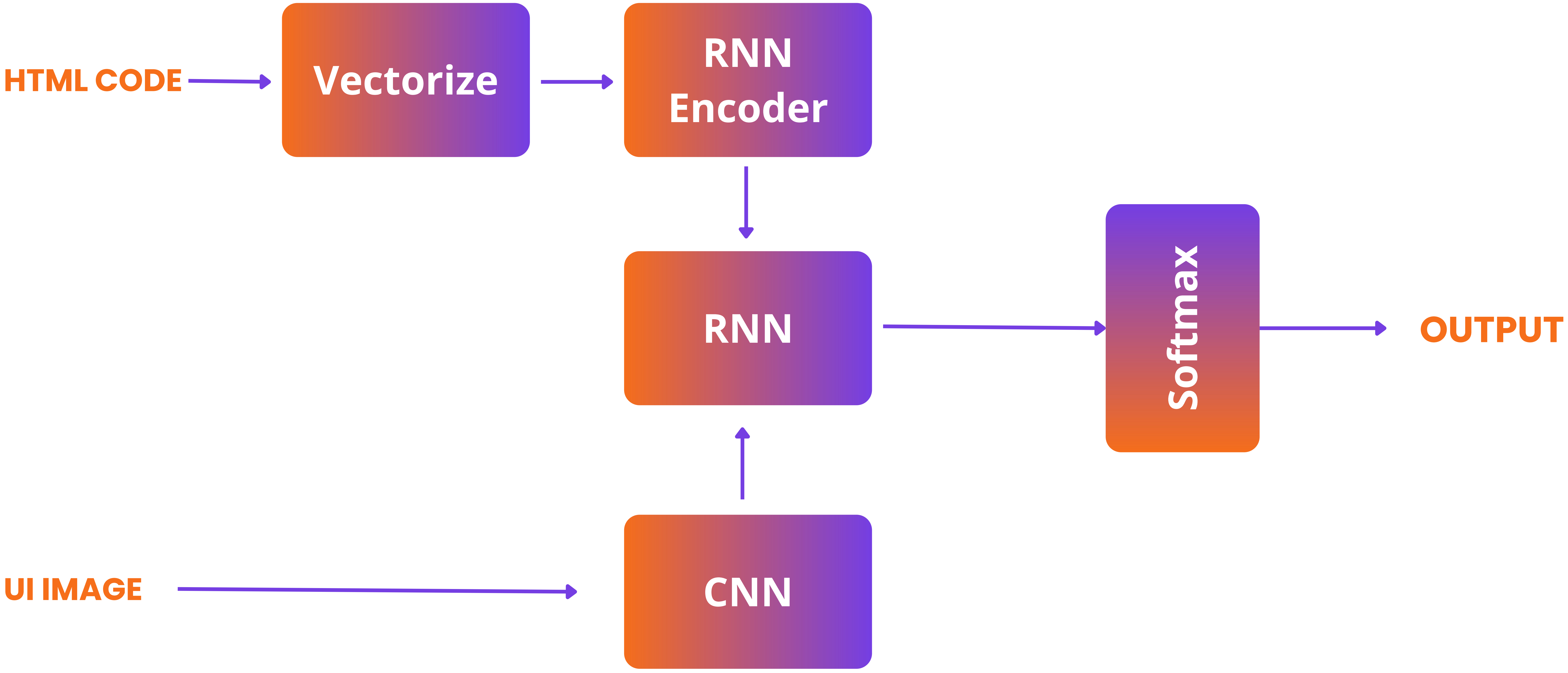
You may need to fine-tune the model by adjusting the hyperparameters or adding more layers to the CNN or RNN model.

08

Deploying the model

The model can be deployed to a production environment where it can be used to convert UI designs to HTML code. We can also create a web application or API that facilitates the same.

Flow of the model



Performance Metrics

Cross-entropy loss, also known as log loss, is a widely used loss function in machine learning and deep learning, particularly in classification tasks.

The formula for cross-entropy loss is:

$$L(y, \hat{y}) = -\Sigma\, y_i \log(\hat{y}_i)$$

where:

- L: the cross-entropy loss
- y: the ground truth label (in this case, the ground truth HTML code)
- $\hat{y}$: the predicted label (in this case, the predicted HTML code)
- y_i: the i-th element of the ground truth label (either 0 or 1, indicating the absence or presence of a specific HTML element or attribute)

$\hat{y}_i$: the i-th element of the predicted label (a value between 0 and 1, indicating the model's confidence that the corresponding HTML element or attribute is present in the predicted code)

Future Scope

01

We can extend this project and enable it to generate new sections or pages of a website based on the previously fed input. without giving any new UI input.

02

We can also apply the dimensionality analysis in this project, i.e. predicting the size of the object in the given image.

03

Various security features along with the code generation can be added in order to make the code generation secure

04

We can research on automating the backend generation and club it together to make a complete automatic website generation platform.

05

We can extend the project further to generate code for more advanced front-end libraries like CSS, JS to implement front-end which contains interactive web pages.

06

We can also modify and extend the model further to include Web3 functionalities and include code generation for libraries like Web3.js and Ethers.js

References

- B. Aşıroğlu et al., "Automatic HTML Code Generation from Mock-Up Images Using Machine Learning Techniques," 2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT), Istanbul, Turkey, 2019, pp. 1-4, doi: 10.1109/EBBT.2019.8741736.
- D. Yashaswini, Sneha and N. Kumar, "HTML Code Generation from Website Images and Sketches using Deep Learning-Based Encoder-Decoder Model," 2022 IEEE 4th International Conference on Cybernetics, Cognition and Machine Learning Applications (ICCCMLA), Goa, India, 2022, pp. 133-138, doi: 10.1109/ICCCMLA56841.2022.9989298.
- B. Aşiroğlu et al., "A Deep Learning Based Object Detection System for User Interface Code Generation," 2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, 2022, pp. 1-5, doi: 10.1109/HORA55278.2022.9799941.
- Generating Smart Contract Front-ends Using Machine Learning Techniques"by X. Li, X. Ma, and C. Zhang
- Automated Smart Contract Front-end Generation by S. Schuster, S. Schneckenburger, and K. Meißner