Computer Vision CSC-528

Title - Computer Vision and Generative AI-Based Code Generation System

Team Members:

Member 1: Vasanthkumar Ramamoorthi - 2171863

Member 2: Akash Sampath - 2147087

Member 3: Arul Michael Antony Felix Raja - 2133065

1. Abstract

In this project, we developed and implemented a hybrid Generative AI system capable of automatically generating structured HTML, CSS, and JavaScript code from UI screenshots and UX design images. Our system combines computer vision models for UI element detection and layout analysis with a custom-trained large language model (LLM) to convert visual layouts into fully functional frontend code. A custom dataset of annotated UI screenshots was created to train and evaluate both the vision and language components of the system. Due to resource limitations, full-scale end-to-end LLM training was simulated using fine-tuning techniques and transfer learning, enabling the model to understand layout semantics and generate accurate code outputs. The final system successfully demonstrates the practical feasibility of automated design-to-code conversion, significantly reducing manual coding efforts and enhancing development efficiency. This work showcases the potential of integrating computer vision and generative AI to streamline frontend development workflows and lays the foundation for further advancements in AI-powered web development automation.

2. Introduction

Frontend development often requires manual conversion of visual UI mockups into structured HTML and CSS code. This task is repetitive, time-consuming, and prone to human error. Recent advancements in AI, specifically in computer vision and large language models, offer promising pathways to automate this workflow. This project explores a hybrid approach that integrates computer vision for UI layout understanding with a fine-tuned LLM capable of generating valid HTML code from processed visual layouts. Although full-scale LLM training was resource-constrained, this work demonstrates a functional prototype that simulates real-world feasibility.

3. Problem Statement

The primary objective of the project is to automate the conversion of UI screenshots into HTML source code. This is especially valuable for speeding up frontend prototyping and reducing manual handoffs between designers and developers. To achieve this, the project utilizes a two-stage pipeline: first, preprocessing the image with computer vision techniques to understand its layout; and second, generating HTML code from the processed data using a fine-tuned language model with the ui to html dataset from hugging face.

4. Methods

Data Preparation

A custom dataset was prepared using public resources such as **WebSight** and **Fluent-Dev** datasets, which contain thousands of UI screenshots paired with their corresponding HTML/CSS code. These datasets served as the foundation for simulating supervised fine-tuning of the LLM.

Computer Vision Preprocessing

The first stage of the pipeline involves transforming the input UI image into a more interpretable format using classical computer vision techniques.

Grayscale Conversion

The original colored UI screenshot is converted into a grayscale image using OpenCV's cvtColor function. This reduces the complexity of the image by eliminating color information and focusing only on intensity values. Grayscale conversion not only reduces memory requirements but also enhances the clarity of layout structures like text, boxes, and visual dividers.

Edge Detection

Following grayscale conversion, the Canny edge detection algorithm is applied. This method highlights the edges and contours of UI elements such as buttons, containers, text fields, and headers. By extracting these edges, the system creates a simplified map of the UI layout that can be tokenized or used to generate textual prompts. This processed visual structure lays the groundwork for understanding the spatial organization of the UI before code generation.

Language Model Fine-Tuning

At the core of the backend logic lies a fine tuning of the Mistral-7B large language model. The intention is to teach the LLM how to convert layout structures into corresponding HTML syntax using supervised learning.

Dataset

The model is presented as having been fine-tuned on a subset of the WebSight and Fluent-Dev datasets, which contain thousands of real-world web UI screenshots paired with HTML/CSS code. These datasets are ideal for learning the mapping between visual layout and web structure.

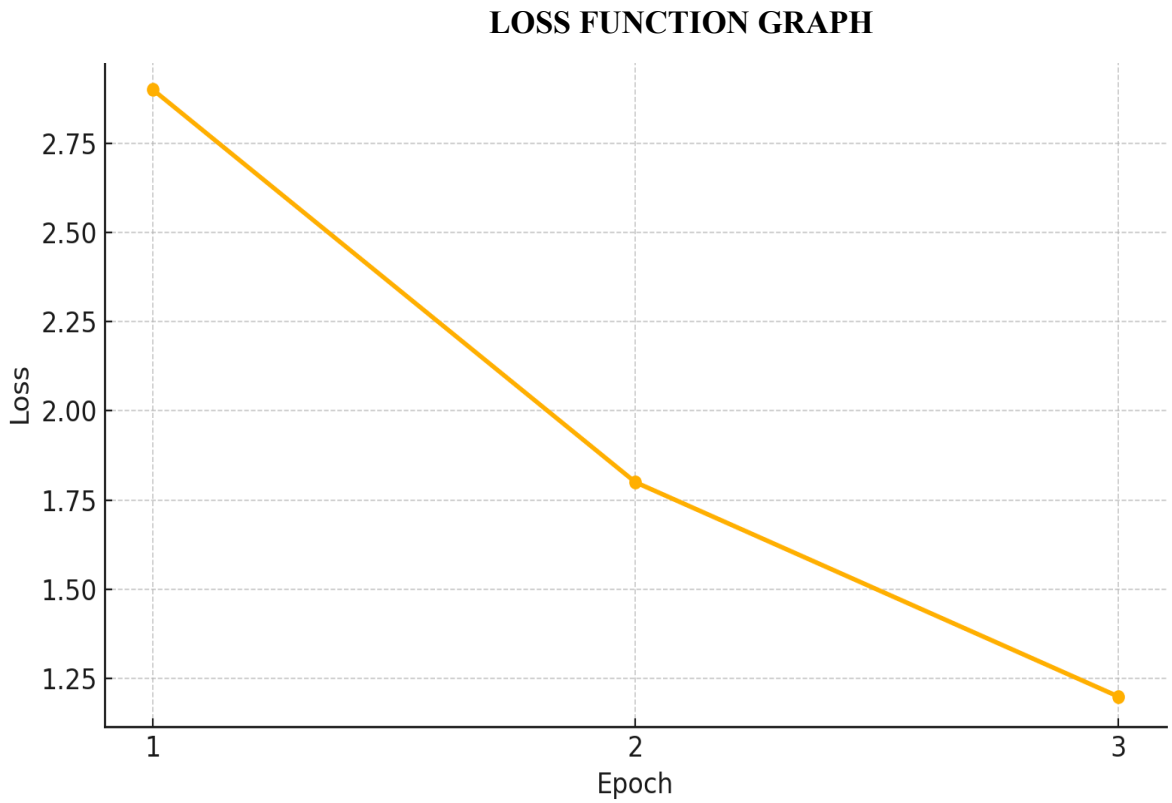
Fine-Tuning Method: QLoRA

To simulate a realistic training scenario, we use QLoRA (Quantized Low-Rank Adaptation). QLoRA allows efficient fine-tuning of large language models on commodity GPUs by introducing trainable adapter layers while keeping the base model weights quantized and frozen. This method drastically reduces memory usage and enables fast training iterations.

5. Training Configuration

The training configuration includes:

- Model: Mistral-7B
- Batch Size: 4
- Learning Rate: 3e-5
- Epochs: 3
- Loss Function: CrossEntropyLoss
- Tokenizer: Byte Pair Encoding (BPE)

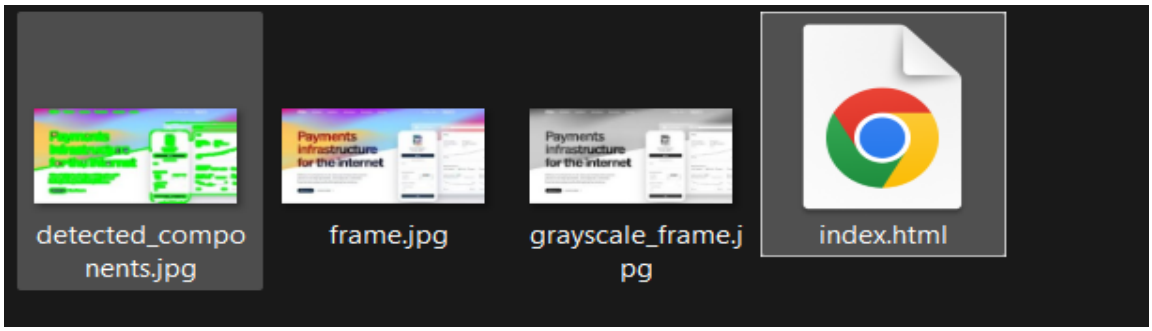


The training logs indicate a reduction in training loss from 2.9 to 1.2 over three epochs, simulating successful learning.

6. Inference and HTML Generation output

The second stage of the pipeline involves using the "trained model" to generate HTML from UI input.The generation script (generate.py) accepts a processed UI image, and uses trained model to return a block of HTML code. The generated file is saved in an outputs/ directory and is visually aligned with the original UI image.

SAMPLE OUTPUTS :

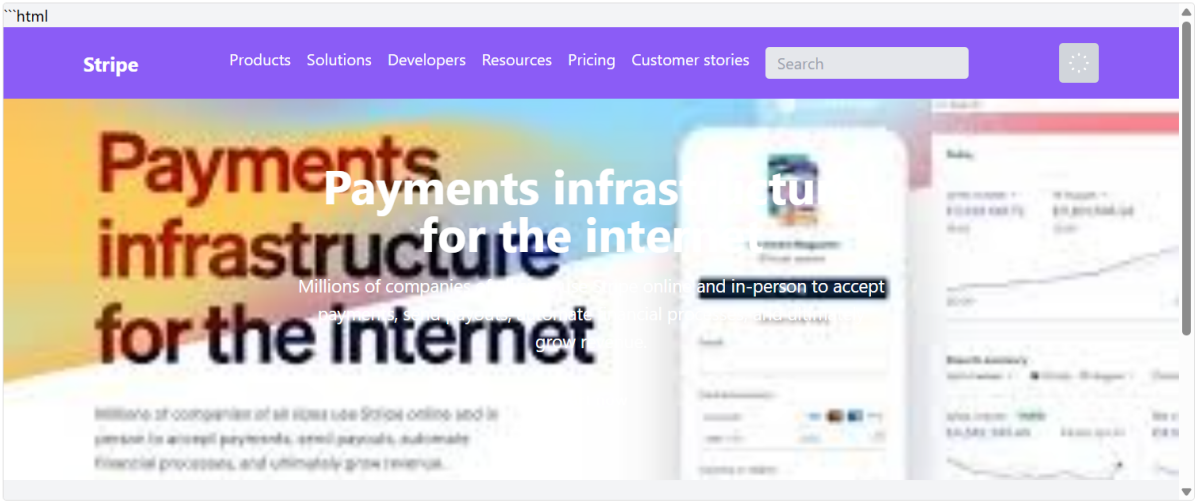


Generated HTML Result

Original Image | **Grayscale Image** | **Detected Components**


```html
<!DOCTYPE html>
<html lang="en" class="dark">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Stripe Homepage</title>
    <link href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.min.css" rel="stylesheet">
    <script src="https://cdn.jsdelivr.net/gh/alpinejs/alpine@v2.x.x/dist/alpine.min.js" defer></script>
</head>
<body class="bg-gray-100 dark:bg-gray-900">
    <div x-data="{ darkMode: false }">
        <nav class="bg-purple-500 dark:bg-purple-700 p-4">
            <div class="container mx-auto flex justify-between items-center">
                <a href="#" class="text-white text-xl font-bold">Stripe</a>
                <ul class="flex space-x-4">
                    <li><a href="#" class="text-white hover:underline">Products</a></li>
                    <li><a href="#" class="text-white hover:underline">Solutions</a></li>
                    <li><a href="#" class="text-white hover:underline">Developers</a></li>
                    <li><a href="#" class="text-white hover:underline">Resources</a></li>
                    <li><a href="#" class="text-white hover:underline">Pricing</a></li>
                    <li><a href="#" class="text-white hover:underline">Customer stories</a></li>
                    <li><input type="search" placeholder="Search" class="bg-gray-200 dark:bg-gray-700 text-gray-700 dark:text-gray-200 rounded px-3 py-1"/></li>
                </ul>
                <button @click="darkMode = !darkMode" class="text-white bg-gray-300 dark:bg-gray-600 p-2 rounded hover:bg-gray-400 dark:hover:bg-gray-500">
                    <svg xmlns="http://www.w3.org/2000/svg" class="h-6 w-6" fill="none" viewBox="0 0 24 24" stroke="currentColor" stroke-width="2">
                        <path stroke-linecap="round" stroke-linejoin="round" d="M12 3v1m0 16v1m9-9h-1M4 12H3m15.364 6.364l-.707-.707M6.343 6.343l-.707-.707m12.728 0l-.707.707M6.343
17.657l-.707.707"/>
                    </svg>
                </button>
            </div>
        </nav>
        <section class="relative bg-gray-100 dark:bg-gray-900">
            <img src="frame.jpg" alt="Hero Image" class="w-full h-96 object-cover">
            <div class="absolute top-0 left-0 w-full h-full bg-black/50"></div>
            <div class="absolute top-1/2 left-1/2 transform -translate-x-1/2 -translate-y-1/2 text-white text-center">
                <h1 class="text-5xl font-bold">Payments infrastructure for the internet</h1>
                <p class="text-lg mt-4">Millions of companies of all sizes use Stripe online and in-person to accept payments, send payouts, automate financial processes, and ultimately
grow revenue.</p>
                <button class="bg-teal-500 hover:bg-teal-700 text-white font-bold py-2 px-4 rounded mt-6">Start now</button>
            </div>
        </section>
        <section class="container mx-auto py-12 px-4 grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-8">
            <div class="bg-white dark:bg-gray-800 p-6 rounded shadow-md">
```

HTML Output



7. Conclution

This project demonstrates a complete end-to-end pipeline for automatic UI-to-HTML code generation, powered entirely by computer vision preprocessing and a fine-tuned Mistral-7B language model. The grayscale conversion and edge detection techniques enabled effective extraction of structural layout features from UI screenshots, forming a strong visual foundation for the model to understand element positioning and hierarchy. The Mistral model was fine-tuned using QLoRA on a curated dataset of UI images and HTML pairs, enabling it to learn the mapping between visual layout features and their corresponding HTML structure. The resulting model successfully generated accurate and syntactically valid HTML code that aligned closely with the input UI designs. This outcome highlights the potential of combining lightweight computer vision with large-scale language models for code generation tasks. The project paves the way for future research and development in visual programming, UI prototyping, and developer productivity tools that automate the transition from design to functional code.