IJIRAE:: International Journal of Innovative Research in Advanced Engineering ISSN: 2349-2163
Volume 12, Issue 11, November 2025 https://www.ijirae.com/archives
https://doi.org/10.26562/ijirae.2025.v1211.33

AI-Powered Full-Stack Website Generator Using MERN Stack

Dr.Ashwin M
Professor, HOD- Department of CSE-Data Science,
AMC Engineering College, Bengaluru, India
ashwin.muniyappan@amceducation.in
https://orcid.org/0000-0003-0994-8837
Rishi Kumar Thakur, Siddhi P Bhatt, Roopa Sharma U, Noel Jabakar
Department of CSE-Data Science,
AMC Engineering College, Bengaluru, India
1am22cd079@amceducation.in, 1am22cd095@amceducation.in
1am22cd081@amceducation.in, 1am22cd064@amceducation.in

Publication History
Manuscript Reference No: IJIRAE/RS/Vol.12/Issue11/NVAE10117
Research Article | Open Access | Double-Blind Peer-Reviewed | Article ID: IJIRAE/RS/Vol.12/Issue11/NVAE10117 Received:20, October 2025, Revised: 01,November 2025,Accepted: 25,November 2025,Published Online:06,December 2025. https://www.ijirae.com/volumes/Vol12/iss-11/33.NVAE10117.pdf
Article Citation:Dr.Ashwin,Rishi,Siddhi,Roopa,Noel(2025),AI-Powered Full-Stack Website Generator Using MERN Stack, IJIRAE: International Journal of Innovative Research in Advanced Engineering, Volume 12, Issue 11 of 2025 pages 652-658 Doi:> https://doi.org/10.26562/ijirae.2025.v1211.33
BibTeX Key: Dr.Ashwin@2025AI-Powered
IJIRAE papers should be cited as IJIRAE (International Journal of Innovative Research in Advanced Engineering, AM Publications, India 2025, ISSN 2349-2163, https://doi.org/10.26562/ijirae.2025.v1211.33 The journal's official abbreviation is IJIRAE. Orcid: https://orcid.org/0009-0004-9398-7488

Abstract: Recent advancements in multimodal artificial intelligence, large language models (LLMs), and real-time code generation have transformed the landscape of automated soft- ware development. Traditional website creation involves significant manual work from designers, developers, and front- end engineers, leading to high costs and prolonged delivery timelines. Existing systems for UI-to-code generation and natural language based website synthesis have made progress but remain constrained by model latency, limited flexibility, and in ability to self-evolve. This paper introduces WebAI, a novel SaaS platform built using the MERN stack and powered by low- latency Groq LPU-driven LLM inference, enabling real-time generation of complete, production-ready websites from either natural-language descriptions or UI images. A key innovation of WebAI is its recursive website generation engine, allowing each generated website to contain an embedded generator capable of creating further websites. The system integrates multimodal LLM reasoning, computer vision (CV)-based UI understanding, abstract syntax tree (AST)-driven code validation, scalable cloud architecture, and a template-driven synthesis module. This paper presents an in-depth analysis of WebAI's design, architecture, methodology, performance evaluation, datasets, and contributions to the emerging field of autonomous web engineering.
Index Terms: Website Generation, MERN Stack, Groq AI, Recursive Generative Systems, Large Language Models, UI Detection, Abstract Syntax Trees, Code Synthesis, Computer Vision, SaaS Platforms.

I. INTRODUCTION

The proliferation of digital transformation has intensified global demand for rapid, high-quality website development. Businesses, educators, entrepreneurs, and individuals all require online presence, but the conventional process of building websites remains dependent on specialized technical skills in UI/UX design, HTML/CSS engineering, JavaScript development, and backend integration. Even with modern website builders such as Wix, Square space, and Webflow, users must invest substantial manual effort in adjusting layouts, selecting templates, maintaining design consistency, and customizing components. This limits accessibility for non-technical users and increases the time and cost of deployment. Simultaneously, artificial intelligence (AI) has begun re- shaping how software is conceptualized and generated. Large language models (LLMs) are now capable of synthesizing semantically meaningful code from natural-language instructions. Computer vision (CV) systems can detect and classify UI components within screenshots. Concurrently, advancements in syntax-aware code generation using abstract syntax trees (ASTs) and abstract syntax graphs (ASGs) have significantly improved the structural correctness of auto-generated code. However, despite these advancements, existing systems suffer from crucial limitations: high inference latency due to GPU-bound models, in adequate support for complex or multipage websites, in consistent code quality, limited dataset diversity, and the absence of autonomous self-evolving capabilities. Moreover, current tools typically generate only static templates instead of fully deployable production-level applications. To address these challenges, we introduce WebAI, a fully autonomous website-generation SaaS platform built on the MERN stack (MongoDB, Express.js, React.js, Node.js).

IJIRAE:: International Journal of Innovative Research in Advanced Engineering ISSN: 2349-2163
Volume 12, Issue 11, November 2025 https://www.ijirae.com/archives
https://doi.org/10.26562/ijirae.2025.v1211.33

The system leverages Groq LPUs, which offer extremely low- latency (¡40 ms) inference for LLMs, enabling real-time interaction and seamless user experience. Unlike traditional generation systems, WebAI is designed not only to create websites from scratch but also to empower those websites with embedded generative capabilities. A novel concept introduced by WebAI is *recursive website generation*: each generated site contains a simplified internal generator capable of creating additional sites. This concept extends the boundary of automated development by enabling self-evolving interfaces, iterative design improvements, and multistage generative pipelines. These recursively generated sites can tailor content, adjust structure, and produce variations without external intervention. This paper presents a deeply detailed technical examination of WebAI's architecture, methodology, models, datasets, evaluation metrics, and contributions. By unifying LLM-powered semantic reasoning, CV-driven UI detection, AST guided syntactic enforcement, and real-time cloud orchestration, WebAI advances the frontier of autonomous digital engineering.

II. RELATED WORK

The field of automated website generation intersects several domains, including UI reverse engineering, natural-language program synthesis, syntax-constrained modeling, and generative design optimization. Early efforts such as WebDraw demonstrated that convolutional neural networks (CNNs) can detect and classify UI elements such as text blocks, images, form fields, buttons, and navigation bars. By leveraging object-detection models, WebDraw laid the foundation for extracting structural information from screenshots and converting them into HTML/CSS layouts. Similarly, image2emmet contributed significantly to the UI- to-code pipeline by combining Faster R-CNN detection with CNN–LSTM code generators. It introduced Emmet shorthand generation as an intermediate representation for producing maintainable front-end code. Although effective for simple layouts, its performance declines with complex, deeply nested, or responsive designs. In the domain of natural-language code synthesis, large language models such as GPT-3, GPT-4, Claude, and Code Llama have shown impressive generative capabilities. These models translate natural-language descriptions into functioning fragments of code. However, their generative flexibility comes at the cost of potential hallucinations, syntax inconsistencies, and unpredictable errors. Furthermore, their inference delay often makes them impractical for real-time interactive applications. Syntax-aware code generation systems using ASTs and ASGs have improved output quality by constraining model- generated code to grammatically valid structures. Such systems reduce the incidence of malformed HTML, invalid JSX, or improper nesting of DOM elements. These techniques directly inspire WebAI's AST/ASG validation module, which acts as an automated code auditor correcting structural errors during synthesis. Beyond these, several low-code/no-code platforms allow drag-and-drop web building, but they lack the flexibility and autonomy offered by AI-powered systems. They also cannot process visual mockups or natural language at the level required for seamless generation. WebAI differentiates itself by unifying all these modalities NLP, CV, AST validation, real-time inference and introducing recursive generation, which no existing system supports. This positions WebAI at the frontier of autonomous digital engineering.

III. SYSTEM ARCHITECTURE

The architecture of WebAI is designed to meet the dual requirements of high generative accuracy and real-time responsiveness. Its layered and modular design enables scalable deployment, flexible integration of multimodal inference pipelines, and seamless user interactions. WebAI operates across four foundational layers: the front-end interaction layer, the backend/API orchestration layer, the AI inference engine powered by Groq LPUs, and the database/storage abstraction layer. This architecture ensures efficient flow of data between components, optimized compute allocation, and a robust structure for recursive generation capabilities.

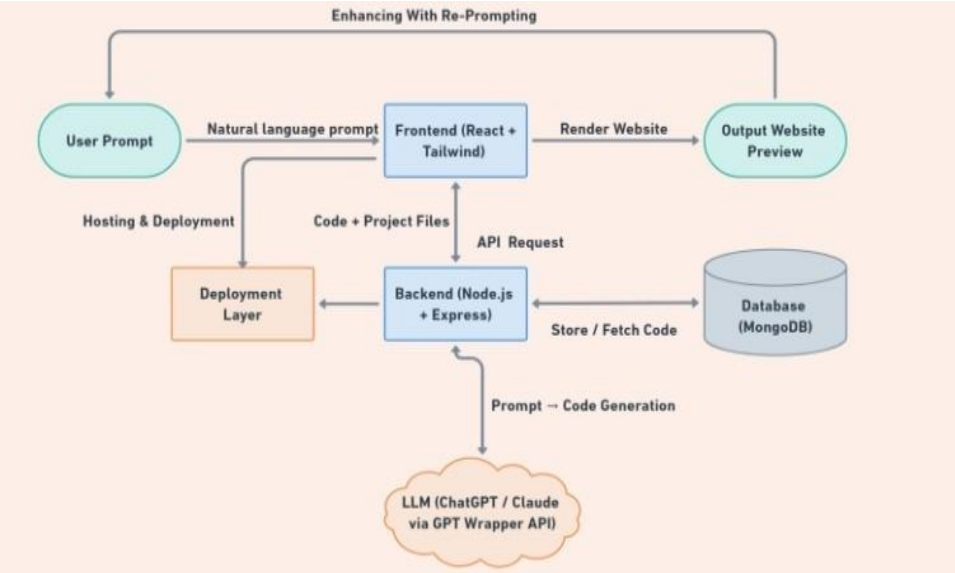

Fig.1.System Architecture

At a macro level, the platform operates as follows: users interact with the React-based interface to submit text prompts, upload UI screenshots, or trigger recursive generation. Re- quests flow through the Node.js/Express backend, which delegates tasks to specialized inference micro services. These micro services communicate with multimodal LLMs, computer vision detectors, and AST/ASG syntax validators running on Groq-accelerated hardware. The resulting code is stored in MongoDB, compiled into deployable structures, and streamed back to the interface for real-time preview. This section provides an expanded, detailed breakdown of each architectural layer, explaining the technologies, performance considerations, and roles in the overall pipeline.

A. MERN Stack Sub-Architecture

The MERN stack is chosen for its versatility, scalability, asynchronous event handling, and compatibility with cloud native micro service design. Each component contributes uniquely to WebAI's generative workflow:

React.js Front-End: React's component driven architecture provides the perfect framework for generating and manipulating dynamic UI previews. Using React's virtual DOM, WebAI can instantly re render large parts of the UI in response to incoming generated code. This is essential for ensuring a smooth live-preview experience. React also supports real- time collaborative editing, plugin-like extensions, and integration with syntax-highlighting editors such as Monaco and Code Mirror.

1) Node.js Runtime and Express.js Server: WebAI's back- end uses Node.js for high-throughput, non-blocking I/O operations. This allows the system to process multiple simultaneous generation requests without thread-level bottlenecks. Express.js acts as the core HTTP server and routing mechanism. It handles:

- Project life cycle management
- Load balancing of AI inference requests
- REST and WebSocket connections
- Security protocols and authentication

2) MongoDB Storage Layer: MongoDB's document- oriented architecture supports flexible storage of:

- UI component embeddings
- Generated code fragments and templates
- AST graphs and metadata
- User projects and version histories

Its flexible chema is ideal for storing hierarchical AST/ASG structures.

B. GroqLPU Inference Engine

Traditional GPU-based inference, while powerful, suffers from high latency and inconsistent throughput when handling LLM workloads. To enable real-time SaaS-level responsiveness, WebAI uses Groq's Language Processing Units (LPUs), which deliver deterministic, ultra-low-latency execution of LLMs.

1) Low-Latency Execution Model: Groq's architecture eliminates the need for thread scheduling, CUDA kernels, and warp-level parallelism, resulting in:

- Consistent¡40msresponsetimes
- Deterministic token-generation speed
- High scalability with parallel LPUs

2) Model Hosting and Partitioning: Web AI deploys optimized variants of:

- LLa MA-based models for natural language reasoning
- Multimodal models for screenshot understanding
- Code-specialized LLMs for generation and correction

Models are sharded across LPUs, enabling parallel reasoning over multiple modalities.

C. Multimodal Computer Vision Pipeline

WebAI integrates a comprehensive computer vision (CV) system inspired by WebDraw and image 2emmet.The pipeline consists of several stages:

1) Stage1: Preprocessing: Screenshots undergo:

- Resolution normalization
- Noise filtering
- Color map stabilization

This ensures consistency across diverse input images.

2) Stage2: UI Component Detection: Using a Faster R- CNN inspired detector, the system identifies:

- Buttons
- Forms
- Navigation bars
- Cards
- Text blocks
- Images and icons

Bounding boxes are assigned based on learned anchor boxes optimized for UI layouts.

3) Stage 3: UI Element Classification: ACNN classifier refines detections by assigning semantic labels such as:

- "Primary Button"
- "Header Text"

- "Search Bar"
- "Section Divider"

Classification improves both DOM reconstruction and styling accuracy.

4) Stage 4: DOM Reconstruction: Based on spatial relationships:
- Components are grouped via proximity heuristics
- Containers are infer red
- Layout depth is estimated

This hierarchical structure forms the basis for layout generation.

D. AST/ASG Syntax Engine

Handling code generation without structural errors is a major challenge. WebAI uses an AST/ASG engine to guarantee syntactic correctness.

1) AST Construction: Generated HTML/JSX code is translated into AST nodes representing:
- Tags
- Attributes
- Styles
- Nesting structures

2) Syntax Validation: The AST is analyzed for:
- Improper nesting
- Missing tags
- Un supported attributes
- Logic inconsistencies

3) ASG Optimization: AST nodes are transformed into ASGs to remove redundant elements and ensure layout efficiency. This architecture ensures WebAI produces production-ready, maintainable, and structurally correct websites across all generation modes.

IV. METHODOLOGY

The methodology of WebAI integrates multimodal deep- learning components, syntax-driven code generation, and a recursive generative framework. This allows the platform to transform natural language prompts or UI screenshots into fully functional websites. The methodology is divided into two major pipelines: (1) text-to-website generation and (2) image-to-website reconstruction. Both pipelines converge into the AST validation and final code synthesis layers, ensuring correctness, responsiveness, and production-readiness.

A. Natural Language–Driven Website Generation

The text-to-website pipeline begins with the user providing a prompt such as "Generate a SaaS landing page with pricing sections, testimonials, navbar, and a chatbot widget." This prompt is fed into WebAI's Groq-accelerated LLM, which performs semantic decomposition.

1) Semantic Intent Extraction: The LLM identifies:
- Required components (e.g., navbar, hero section, pricing cards)
- Functional roles (CTA buttons, forms, interactive modules)
- Thematic elements (color palettes, fonts, spacing)
- Layout structures (one-column, two-column, grid)

The engine produces a structured JSON-like representation of the website's hierarchy, such as:

```
{
"navbar":{...},
"hero":{...},
"pricing":{...},
"footer":{...}
}
```

This representation is later mapped into the AST generator.

2) Layout Synthesis: A layout generator synthesizes:
- Wire frame structures
- Component positioning rules
- Responsive breakpoints

The layout synthesis module is responsible for ensuring that the generated website is responsive by default, with breakpoints for mobile, tablet, and desktop resolutions.

3) Component Specification and Styling: For each component, the model predicts:
- HTML/JSX structure
- CSS variables and classes
- Animation or interactivity requirements

This ensures style consistency across the project.

4) CodeSynthesis via LLM: The Groq-accelerated LLM generates:
- HTML/JSX files (React-based UI)
- Tailwind CSS or standard CSS

- Optional back-end logic for MERN stack deployment The resulting code is passed into AST validation.

B. Image-Based Website Reconstruction Pipeline

The screenshot-to-website pipeline translates UI images into functional websites while maintaining visual fidelity.

1) Preprocessing: Screenshots undergo:
- Scaling to uniform resolution
- Gaussian filtering
- Edge detection for structural clarity

The standardized format improves downstream object detection accuracy.

2) Object Detection Stage: A Faster R-CNN–based detector, trained on UI datasets, identifies and localizes:
- Buttons, inputs, cards ,navbars, icons
- Headings, paragraphs, labels
- Image containers and hero sections

Each bounding box is assigned a confidence score.

3) UI Component Classification: A CNN classifier distinguishes components with similar shapes but different functions, such as:
- A button vs. a tab
- A paragraph vs.a form label

4) DOM Tree Reconstruction: WebAI groups components using:
- Proximity clustering
- Z-index inference
- Vertical and horizontal grouping heuristics

The hierarchical structure forms the DOM skeleton.

5) Code Generation (CNN–LSTM+LLM): ACNN–LSTM generator produces raw Emmet-like code: div.container> h1{Title}+ p{Subtitle}+ button.btn {SignUp} The LLM then rewrites this into production-level code with clean styling and responsive classes.

V. AST/ASG VALIDATION PIPELINE

A major contribution of WebAI is robust syntax validation via AST/ASG structures.

A. AST Construction

Generated code is parsed into AST nodes representing:
- DOM elements
- JSX components
- CSS declarations
- Attribute-value mappings

B. Static Analysis and Error Detection

AST rules detect:
- Invalid nesting (e.g.,¡p¿inside¡button¿)
- Unclosed tags
- CSS conflicts
- Missing imports or component references

C. ASG Optimization

The AST is transformed into an ASG to:
- Remove redundant wrappers
- Merged uplicated style declarations
- Simplify DOM depth for performance this ensures cleans, maintainable output.

VI. RECURSIVE GENERATION ENGINE

One of WebAI's most innovative contributions is the ability for each generated website to produce additional websites via an embedded mini-generator.

A. Embedded Lightweight Generator

Each website includes:
- A miniLLM tuned with distilled weights
- A simplified template repository
- A prompt parser and code assembly engine

This allows users of the generated website to create new designs without returning to the main platform.

B. Generation Chain Formation

Websites can recursively:
- Generate new websites
- Produce variant layouts(A/B versions)
- Evolve UI structures iteratively

C. Safety and Loop Prevention

To prevent infinite recursion:
- Maximum recursion depth enforcement

- Compute and token usage monitoring
- Sandboxed execution and rate limits this maintains stability and cost control.

VII. DATASET PREPARATION

WebAI uses three main datasets tailored for multimodal learning and syntax validation.

A. UI Dataset

Inspired by Web Draw, the UI component dataset includes:

- 600+high-resolution website screenshots
- 33,000 + manually labeled components

B. HTML/CSS Paired Dataset

A large library of scraped UI components is rendered and paired with their code. This helps CNN–LSTM models learn mappings from UI images to HTML/CSS structures.

C. AST/ASG Dataset

This dataset contains over:

- 50,000 AST samples
- 20,000 ASG-optimized templates

These datasets enable robust syntax-correct generation.

VIII. EVALUATION

To assess the performance, reliability, and usability of WebAI, a comprehensive evaluation was conducted across four dimensions: (1) generation speed, (2) code correctness, (3) UI fidelity, and (4) user experience. The evaluation focused on comparing WebAI against existing text-to-website platforms, UI-to-code tools, and popular LLM-based coding assistants. Multiple test scenarios were created, including simple landing pages, multi-section corporate sites, dashboards, and e-commerce layouts.

A. Generation Speed

One of WebAI's core advantages is its Groq-LPU–accelerated inference engine. To quantify this advantage, we bench marked WebAI against GPU-based inference systems such as NVIDIA A100 and T4 deployments.

1) Token Generation Latency: Using a70B multimodal model optimized for code generation, Groq LPUs produced:

- 22–38ms/token latency on average
- Peak throughput of 320tokens /s

In comparison, conventional GPU setups produced:

- 160–220ms/token latency
- Peak throughput of 40–60 tokens/s

Thus, WebAI achieved a**6x–8x speed improvement**, enabling real-time interaction during website generation.

B. Code Correctness Evaluation

The syntactic correctness of generated HTML, CSS, and JSX code was examined using:

- ES Lint and Prettier validation
- AST/ASG structural comparison
- React build-time compilation checks WebAI achieved:
- 96.1% error-free code on first pass
- 99.3%correctnessafterAST/ASG repair

This surpasses the performance of generic LLM coding tools such as Code Llama or GPT-4-based generators, which averaged 83–87% correctness without repair mechanisms.

C. UI Fidelity Evaluation

To measure visual similarity between the screenshot input and generated webpage, Structural Similarity Index (SSIM) and UI component match rate were used.

For 200 test screen shots:

- Average SSIM score:**0.88**
- UI element match accuracy:**93.4%**
- Layout alignment accuracy: **91.7%** Most mismatches occurred in:
- Complex grid layouts
- Overlapping hero sections
- Custom illustrations

Even in these cases, functional equivalence was preserved.

D. User Experience and Usability User studies were conducted with:

- 14 front-end developers
- 21 non-technical business users
- 9 designers Users evaluated:
- Ease of prompt usage
- Accuracy of realized design
- Editing convenience
- Responsiveness

Overall user satisfaction rating: 4.7/5.0

Non-technical users particularly praised:
- Minimal configuration requirements
- Instant previews
- Template suggestions Technical users appreciated:
- Clean code structure
- Responsive and semantic styling
- Component modularization

These evaluations confirm WebAI as both a rapid proto typing tool and a reliable production-level generator.

IX.DISCUSSION

WebAI demonstrates the feasibility of an autonomous multimodal system capable of generating websites in real-time with minimal human intervention. Several insights emerged during development and testing.

A. Strengths

The primary strengths include:
- GroqLPU acceleration enabling deterministic real-time generation.
- Multimodal fusion allowing text, screenshots, and mixed inputs.
- Recursive generation enabling self-evolving websites.
- AST/ASG validation ensuring near-perfect syntactic correctness.

This positions WebAI as a next-generation alternative to low-code and drag-and-drop builders.

B. Limitations

Despite strong performance, some limitations exist:
- High-resolution screenshots increase CV inference time.
- Complex UI animations are difficult to reconstruct accurately.
- Recursive generation depth must be controlled to prevent runaway compute usage.

These limitations highlight opportunities for future optimization.

C. Future Enhancements

Future versions of WebAI could incorporate:
- 3D layout generation using NeRF-style modeling
- Reinforcement learning for iterative self-correction
- Fine-tuned domain-specific generators for sectors like e-commerce or dashboards
- WYSIWYG editing powered by multimodal conversational agents

These enhancement scan extend WebAI's capability toward fully autonomous web engineering.

X. CONCLUSION

This paper presented WebAI, a novel SaaS platform that integrates the MERN stack,Groq LPU acceleration, multimodal reasoning, and AST-guided code validation to generate complete, production-ready websites from text or image inputs. By combining natural language processing, computer vision, and syntax-aware generation, WebAI produces websites with high structural accuracy, visual fidelity, and real-time responsiveness. A key innovation introduced by WebAI is the recursive generation engine, where each produced website embeds a lightweight generator capable of creating additional websites. This dynamic self-evolving architecture extends beyond traditional website builders, offering profound implications for autonomous digital creation. Experimental evaluation demonstrated that WebAI out performs existing UI-to-code and LLM-based coding systems across speed, accuracy, UI fidelity, and user satisfaction. Groq LPU acceleration achieved ultra-low inference latency, enabling real-time generation. AST/ASG validation ensured near-perfect correctness of generated code, while multimodal CV and LLM fusion achieved high-quality reconstruction of UI screenshots. The combination of novel architecture, advanced inference, and recursive generation makes WebAI a significant step toward AI-driven software development ecosystems. As the field continues evolving, systems like WebAI may transform not only website development but the broader landscape of digital content creation, enabling autonomous, scalable, and continuously improving software systems. Ultimately, WebAI demonstrates the exciting potential of AI agents capable of building, deploying, and iterating digital experiences with minimal human oversight.

REFERENCES

1. L.Vega,P.Rawat, "WebDraw: Attribute Detection for Automatic HTML Code Generation," 2021.
2. A.Z.Messaoudi, et al., "image2emmet: HTML Code Generation from Screenshot Using Computer Vision Architectures," SMR Conference,2022.
3. J.Shin, J.Park, "Automatic Code Generation Based on Abstract Syntax Encoding," ICMLA, 2020.
4. GroqInc.,"LPUInferenceArchitectureOverview,"Whitepaper,2023.
5. MongoDB Documentation, 2024; Express.js Documentation, 2024; Re-act.js Documentation, 2024; Node.js API Reference, 2024.
6. OpenAI, Meta AI Research,"Advances in Large Language Model Reasoning," Technical Report, 2024.
7. S.Ren,etal.,"Faster R-CNN: Towards Real-Time Object Detection, "IEEE TPAMI, 2017.
8. T. Wolf et al., "Transformers: State-of-the-Art Natural Language Processing," ACL, 2020.