2025 6th International Conference on Machine Learning and Human-Computer Interaction (MLHMI) | 979-8-3315-3573-5/25/$31.00 ©2025 IEEE | DOI: 10.1109/MLHMI66056.2025.00023

From Framework to Functionality: Exploring JavaScript and AI Integration in Automated Coloring

Cheng-Yuan Ho
Dept. of Information Management
National Taiwan University
Taipei, Taiwan
tommyho@ntu.edu.tw

Yu-Wen Gong
Dept. of International Business
National Taiwan University
Taipei, Taiwan
b09704048@ntu.edu.tw

Kai-Syun Chen
Dept. of Accounting
National Taiwan University
Taipei, Taiwan
b10612035@ntu.edu.tw

Su Lee
Dept. of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan
b09902082@ntu.edu.tw

Yu-Ting Gong
Dept. of International Business
National Taiwan University
Taipei, Taiwan
b09704010@ntu.edu.tw

Wei-Han Chen
Dept. of Economics
National Taiwan University
Taipei, Taiwan
b12303134@ntu.edu.tw

Wei-An Chen
Dept. of Business Administration
National Taiwan University
Taipei, Taiwan
b10701131@ntu.edu.tw

Abstract—**This article presents an automated coloring system that synergizes JavaScript and Artificial Intelligence (AI) technologies to enhance user experience, drive educational innovation, and improve creative efficiency. The implementation details include using the Canvas element and the Huemint color model to achieve AI coloring, covering steps such as loading layer data, sorting block sizes, monitoring mouse/touch inputs, tool selection, and coloring logic. Additionally, the study outlines appropriate color models for various scenarios and evaluates the advantages and limitations of different techniques. The results demonstrate how a resource-efficient approach can integrate JavaScript and AI technologies to deliver a flexible and efficient automated coloring solution.**

Keywords—JavaScript, AI Coloring, Generative Adversarial Networks (GANs), Canvas, Huemint

I. Introduction

Advancements in technology and the proliferation of digital tools have driven traditional coloring books and picture books toward digital transformation. In education and entertainment, digital tools not only provide richer interactive experiences but also enhance learning outcomes and foster creativity. While current AI digital coloring tools exhibit some degree of automation, significant room for improvement remains in usability, convenience, assistance capabilities, and creative potential. The motivation behind this article stems from three main objectives:

1. Enhancing User Experience: Existing tools with intelligent features lack sufficient assistance, limiting their ability to offer an intuitive and enjoyable creative process.

2. Driving Educational Innovation: In educational contexts, current AI-based coloring systems often fail to provide realistic or vivid results. Improved AI techniques can help students complete tasks quickly and accurately while assimilating knowledge from illustrated materials.

3. Boosting Creative Efficiency: For adolescents with limited leisure time, manual coloring is often time-consuming and labor-intensive. AI automated coloring can significantly enhance creative efficiency, enabling productive use of recreational time for learning, play, and creation.

To address these needs, this article describes the development of an automated coloring system combining JavaScript and AI technologies [1]. The system provides an intelligent coloring experience and fosters innovation in digital coloring tools. It also delves into the technical choices, implementation strategies, and challenges faced during development, offering valuable insights for researchers and practitioners in related fields.

II. Existing AI Coloring Technologies Review

A. Overview of Techniques

Recent advances in AI coloring owe much to deep learning (DL) and generative adversarial networks (GANs). The following outlines six prevalent techniques, along with their strengths and limitations:

1. Traditional Manual Coloring: Relies heavily on user input for selecting and applying colors to image regions. While straightforward, this approach demands significant time and effort, especially for intricate designs, and lacks intelligent assistance.

2. Rule-Based Coloring: Early automated techniques used predefined rules to assign colors based on lines and regions. These methods are inflexible, struggling to handle the complexity and diversity of real-world images.

3. DL Models: Convolutional Neural Networks (CNNs) enable models to learn color patterns from large datasets and apply them to new grayscale images, enhancing the quality and realism of results [2, 3].

4. GANs: GANs utilize paired CNNs, one as a generator and another as a discriminator, to produce high-quality colorized images. They excel at capturing fine details such as textures and subtle color variations [2, 3].

Authorized licensed use limited to: National Taiwan University. Downloaded on August 20,2025 at 13:52:32 UTC from IEEE Xplore. Restrictions apply.

5. Semi-Automated Tools: These tools integrate manual annotations with AI-generated coloring, balancing user creativity with improved efficiency and results.

6. Commercial Software and Open Source Projects: Tools like Adobe Photoshop, DeOldify, and Image Colorization leverage DL to achieve high-quality results. These tools, widely available on platforms like GitHub, demonstrate AI's potential in automated coloring [4].

B. Limitations and Challenges

Despite significant progress, AI coloring technologies face four primary challenges [5-11]:

1. Color Accuracy: Models sometimes fail to reproduce realistic colors, resulting in unnatural color distributions.

2. Contextual Understanding: Current models lack semantic and contextual awareness, underperforming in complex scenes.

3. Dependence on Training Data: Performance hinges on the quality and diversity of training datasets, which may introduce biases.

4. Resource Constraints: DL models demand substantial computational resources, posing challenges for small teams or individual developers.

This article aims to address these challenges, proposing solutions tailored to resource-limited developers and enhancing accuracy and intelligence in automated coloring systems.

III. Proposed Methodology and Implementation

This article adopts a comparative analysis method, collecting existing JavaScript coloring technologies, comparing the advantages and limitations of each, and then developing customized programs for the picture book system website described in this article, for reference by those who need to create coloring web pages. To comprehensively discuss the programming logic, this article will start with how to achieve coloring effects and describe how it combines with AI coloring methods. Finally, it presents the advantages, challenges, and scenarios suitable for the existing technologies and the methods described in this article, hoping that future researchers can continue to deepen their understanding and improve AI coloring technology.

A. Conceptual Framework

The system defines "coloring rules" to ensure logical and visually appealing results. Four key layers guide the process:

1. Outline Layer: As shown in Fig. 1, defines boundaries of different color regions. Non-colorable areas are filled, while transparent sections represent paintable regions visible to users.

2. Color Block Layer: Specifies uniform color blocks. Transparent areas are non-paintable, while colored sections define paintable regions such as hair or clothing.

For example, as shown in Fig. 2, consider a "person" where the hair is assigned as blue and the clothing as red in the color block layer. When the user clicks on the hair, all blue areas on the canvas will be replaced with the selected brush color, effectively targeting the hair region. This approach allows different areas to be designated as separate blocks for applying distinct colors, as illustrated in Fig. 3, where the hair and clothing are treated as separate blocks.

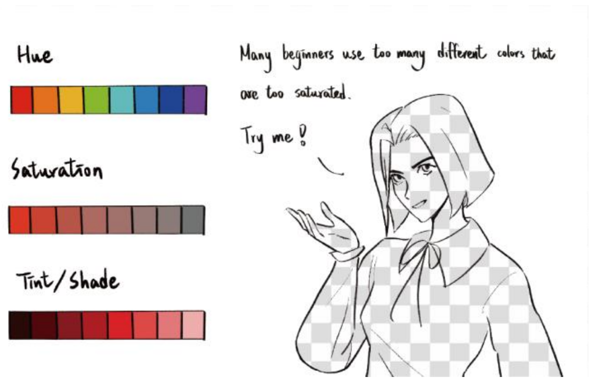

Hue
Saturation
Tint/Shade
Many beginners use too many different colors that are too saturated.
Try me!

Fig. 1. Transparent regions represent areas available for coloring. In this example, the subject is a "person."

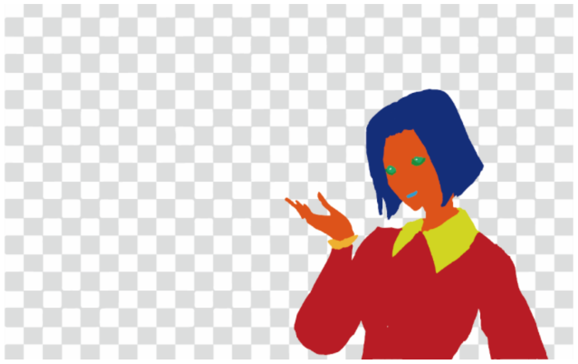


Fig. 2. Defined color blocks and their corresponding boundaries.

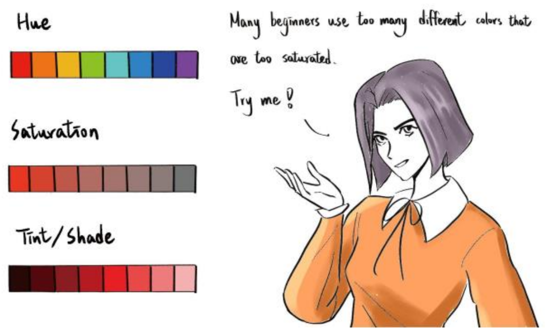

Hue
Saturation
Tint/Shade
Many beginners use too many different colors that are too saturated.
Try me!

Fig. 3. Using the color block layer to differentiate the colors of hair and clothing.

3. Grayscale Layer: Provides shading information, enabling rendering of light and shadow for realistic results.

For instance, as shown in Figures 4 and 5, consider a "person" where the clothing is represented as a single color in the color block layer. By overlaying the grayscale layer and adjusting the shading intensity, the rendering reflects the corresponding light and shadow positions from the grayscale layer. This process gives the clothing a realistic appearance

Authorized licensed use limited to: National Taiwan University. Downloaded on August 20,2025 at 13:52:32 UTC from IEEE Xplore. Restrictions apply.

with highlights and shadows, creating the effect of reflection and depth.

4. Fixed Color Layer: Designed specifically for AI-assisted auto-coloring scenarios. As previously mentioned, the goal of this system is to combine education with entertainment. In certain cases, specific colors have a definitive answer; for example, zinc burns with a yellow-green flame. In such instances, the user must paint the flame yellow-green to achieve the correct result. Other areas require colors to remain reasonable, such as human skin tones, which are typically close to orange-white or brown.

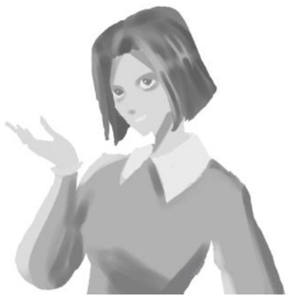
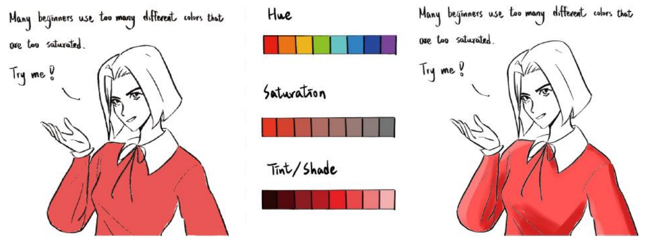


Fig. 4. Illustration of color gradation within a single block.


Many beginners use too many different colors that are too saturated.
Try me!
Hue
Saturation
Tint/Shade
Many beginners use too many different colors that are too saturated.
Try me!
Fig. 5. Comparison of results without (left) and with (right) the grayscale layer. The right-hand image demonstrates enhanced depth and shading.

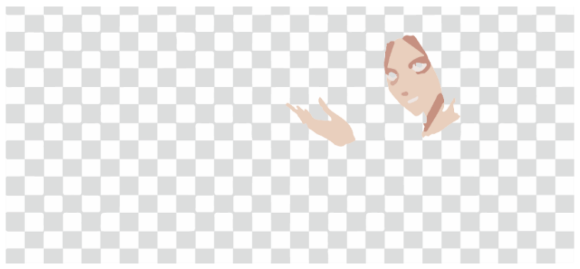
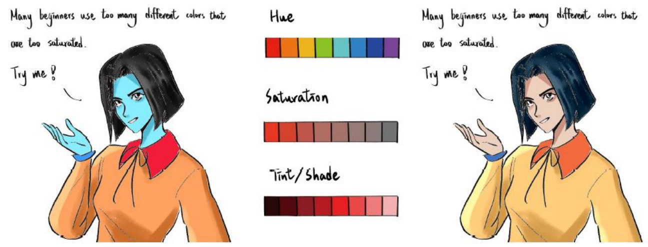


Fig. 6. Example of providing reasonable answers for AI-assisted coloring.


Many beginners use too many different colors that are too saturated.
Try me!
Hue
Saturation
Tint/Shade
Many beginners use too many different colors that are too saturated.
Try me!
Fig. 7. Comparison of results without (left) and with (right) the fixed color layer. The right-hand image shows more realistic skin tones.

Unrestricted AI auto-coloring might produce visually appealing results, but they may not be accurate or reasonable, thereby losing the intended educational value, as illustrated in Figures 6 and 7. Furthermore, transparent areas in the fixed color layer indicate unspecified colors, while colored sections assign their respective colors directly to the corresponding areas on the canvas displayed to the user. This ensures that the resulting color scheme remains logical and appropriate.

B. *Implementation Steps*

The following outlines the implementation steps of the proposed automated coloring system:

1. Loading Layer Data: Upon page load, the system initializes by loading the outline layer, color block layer, grayscale layer, fixed color layer, and edited result layer. The RGB values of all pixels in these layers are extracted using the Canvas element. These layers remain unchanged and serve as references for the coloring logic. The interaction process between the frontend (webpage) and backend (MySQL) is illustrated in Fig. 8. The system converts layer group URLs from HTTP format to Base64 format and binds the converted URLs to corresponding commands, such as image.src. This conversion is necessary to prevent CORS errors, even though Base64 URLs are significantly larger (approximately 80,000 characters). To optimize storage, the URLs are maintained in HTTP format in the database and converted to Base64 only when transmitted to the frontend.

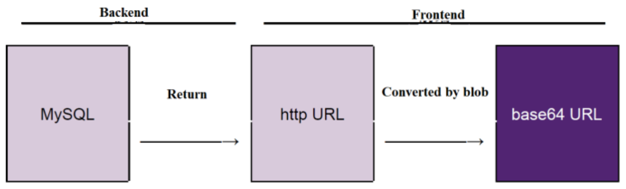
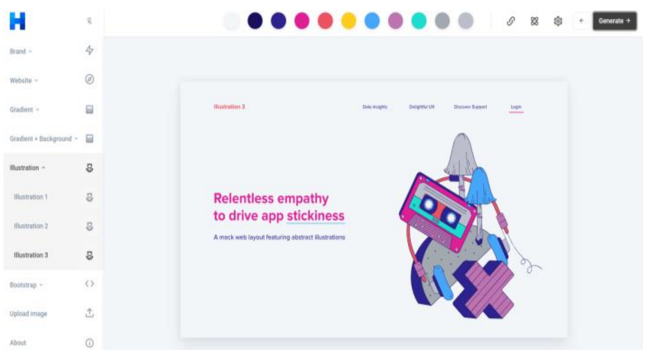


Fig. 8. Process of converting layer URLs to base64 format.


Generate
Relentless empathy to drive app stickiness
A mock web layout featuring abstract illustrations
Fig. 9. A sample diagram showcasing the first color, where blue and white are used for the background.

2. Sorting Block Sizes: Based on the color block layer data, the system internally sorts block sizes to prevent color overlaps caused by sequence issues during the coloring process. The implemented AI color model, Huemint, uses a palette of 12 color codes. Fig. 9 provides an example illustration of this process.

3. Monitoring Mouse/Finger Clicks: When the user clicks on the canvas, the system determines the fill area based on the

Authorized licensed use limited to: National Taiwan University. Downloaded on August 20,2025 at 13:52:32 UTC from IEEE Xplore. Restrictions apply.

color block layer's color information at the clicked position. As shown in Fig. 10, the system calculates the difference between the x/y coordinates of the click and the canvas's left/top boundaries, converts these differences into proportional values relative to the canvas's dimensions, and uses these proportions to determine the target pixel's position. The target pixel's color in the color block layer is then identified.

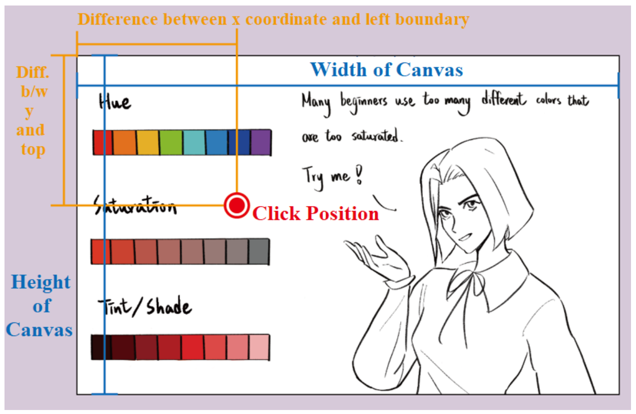

Difference between x coordinate and left boundary
Diff. b/w y and top
Width of Canvas
Height of Canvas
Hue
Saturation
Tint/Shade
Click Position
Many beginners use too many different colors that are too saturated.
Try me!

Fig. 10. Calculation of the x and y coordinates of a clicked point, along with related positional data.

4. Determining the Tool Used: The system identifies the coloring tool selected by the user, such as brush, eraser, AI magic wand, or canvas reset. It further determines whether to use solid fill, gradient fill, or pattern fill.

5. Processing Coloring Logic: The system applies or removes colors based on the selected tool and the sorted block sizes, ensuring no unintended overlaps occur during the coloring process.

6. Saving the Image: After editing is complete, the system converts the final result into a PNG file and uploads it to the Bytescale storage platform. The generated HTTP URL is then sent to the backend for storage, as shown in Fig. 11. To conserve storage space, the system deletes the previous file before uploading a new one.

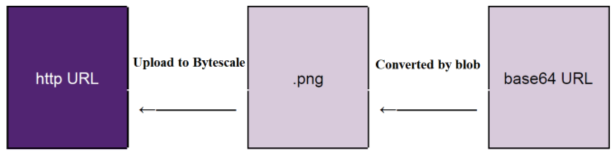


Fig. 11. Image URL conversion process, resulting in an HTTP format URL.

C. Integration of AI Coloring Techniques

This subsection elaborates on the integration of AI coloring techniques into the system [12-21].

i. Color Model Selection

To implement the coloring logic, the system adopts the Huemint color model. Huemint [5, 22] is an automated color-palette generation tool powered by GANs, capable of producing diverse and harmonious color schemes based on input palettes. The application steps are as follows:

1. Importing the Huemint Model: Upon page load, the system imports the Huemint model to generate the initial color palette.

2. Generating the Initial Palette: The system generates the initial color palette using the Huemint model when the user first clicks on the canvas. As shown in Fig. 9, the generated palette is stored in the color block layer for subsequent reference during the coloring process.

3. Dynamic Palette Generation: As users select different regions or tools, the system dynamically generates updated color palettes, ensuring consistent and visually appealing results.

ii. Integration with Coloring Logic

The system combines the Huemint model's generated color schemes with the previously described coloring logic to ensure both accuracy and aesthetics in color application. The integration steps are as follows:

1. Applying Color Schemes: When determining the fill area, the system references the Huemint-generated palette to select the appropriate fill color.

2. Dynamic Adjustments: Based on user actions, the system dynamically adjusts the palette, ensuring that the applied colors remain harmonious and varied.

3. Optimizing Fill Effects: By leveraging information from the sorted block size layer, the system optimizes fill effects, preventing errors such as unintended color overlaps.

IV. Discussion on Research Methods

As previously described, this article does not aim to optimize AI models themselves but focuses on establishing game rules and leveraging algorithms to achieve appropriate outcomes. This approach is ideal for beginners learning and understanding AI coloring systems. It is also suitable for small teams or individuals seeking to implement AI coloring functionality with limited computational resources, eliminating the need to train AI models independently. The following discussion compares this approach with other coloring techniques, including JavaScript-based coloring technologies and existing AI coloring methods [1-22].

A. Comparison with JavaScript-Based Coloring Techniques

Existing JavaScript-based coloring techniques can be categorized into two primary approaches:

1. HTML Element-based Coloring (HEC): This method involves placing HTML elements such as canvas, SVG, div, and fabric.polygon, underneath transparent sections of an image that define colorable regions. When the user clicks on a colorable area, the corresponding HTML element is filled with the selected color.

- Advantages: Simple implementation and easy maintenance.
- Limitations: Difficult to handle irregularly shaped regions and cannot produce natural shading or reflective effects.

Authorized licensed use limited to: National Taiwan University. Downloaded on August 20,2025 at 13:52:32 UTC from IEEE Xplore. Restrictions apply.

2. Different Alpha Values for Colorable Blocks (DAC): This method stores each irregular colorable region as a separate PNG image, assigning distinct alpha values to the pixels of each region. When the user clicks on a colorable area, pixels with the same alpha value as the clicked position are filled with the chosen color. The advantage of this method is convenient for designers to access and manipulate layer images while the limitations are difficult to accurately position layers, less flexible programming, and longer image loading times.

The proposed method in this article converts colorable regions in images into a color block layer, utilizes an AI model for palette generation, and automates the coloring process. This approach effectively handles complex shapes and numerous colorable regions with high precision and natural coloring effects, though creating the layers can be labor-intensive.

The characteristics, advantages, disadvantages, and applicable scenarios of these three techniques are summarized in Table 1.

TABLE I. THREE JAVASCRIPT-BASED COLORING TECHNIQUES COMPARISON

Feature	HEC	DAC	This article
Boundary alignment	Difficult	Easy	Easy
Color filling	Easy	Moderate	Difficult
Layer import time	No layers required	Variable	Regularized
Color layering	Flat or unnatural layers	Can include shadows and highlights	Can include shadows and highlights
Advantages	Simple code, easy to maintain	Convenient for storing layer images	Handles complex images with many adjacent regions
Disadvantages	Cannot conform to irregular shapes	Challenging to position layers; lacks flexibility	Labor-intensive layer creation
Suitable scenarios	Large, dispersed coloring areas	Fewer, simpler coloring regions	Many adjacent coloring regions

B. Comparison with Existing AI Coloring Techniques

The key differences for coloring between the direct training GANs (DTG) and the indirect approach of using a pre-trained AI model (IPA), as employed in this article, lie in implementation complexity, coloring speed, and color accuracy:

1. DTG: This approach requires feeding extensive datasets to train an AI model, a process that is both time- and resource-intensive. Coloring speed depends on the model, typically taking 3-8 seconds. Color accuracy is influenced by the quality of the model training but tends to be lower for non-realistic images.

2. IPA: This article utilizes the Huemint API to generate a palette, which is then applied to fill colors sequentially. This approach is easy to implement and execute, with the API returning palette results in approximately 8 seconds. Also, it has a higher accuracy that means it has high precision for areas specified by the fixed color layer, but lower accuracy in other regions.

The characteristics and suitable scenarios for these two AI-based techniques are compared in Table 2.

TABLE II. TWO AI COLORING TECHNIQUES COMPARISON

Feature	DTG	IPA (this article)
Implementation Complexity	High, requires significant time and resources for training	Low
Coloring Speed	Approximately 3–8 seconds	Approximately 8 seconds
Color Accuracy	Medium to Low	High to Low, depending on region-specific color assignments
Available Color Count	Unlimited	Up to 12 colors
Model Dependency	High	Medium
Suitable Scenarios	- Resource-rich teams - Images with multiple colorable regions - Availability of sufficient training data in the same style	- Small teams or individuals - Fewer colorable regions - Unique artistic styles with limited existing data

From Tables 1 and 2, it can be concluded that the proposed method in this article is well-suited for handling images with complex shapes and numerous adjacent colorable regions. It achieves flexible and natural coloring while retaining shading and reflective effects.

V. CONCLUSION

By integrating JavaScript and AI technologies, this article develops an efficient, high-quality, and intelligent automated coloring system. The proposed system enhances user experience, drives educational innovation, and improves creative efficiency.

Future work will address the following areas for optimization:

1. Palette Size Limitations: Expanding the number of colors provided by the AI model.

2. Blending Issues at Adjacent Block Boundaries: Refining the handling of color transitions.

3. Context-Aware Optimal Color Selection: Enhancing adaptability based on the scene.

These improvements aim to increase the accuracy and intelligence of the AI model, explore broader application scenarios, and achieve deeper integration with other technologies to deliver an even better coloring experience for users.

REFERENCES

[1] Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep Learning. MIT Press.

[2] Radford, A., Metz, L., and Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434.

[3] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84-90.

[4] Awesome Software for Image Coloring, [Online] Available: https://github.com/oskar-j/awesome-image-coloring

[5] Johnson, J., Alahi, A., and Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. arXiv preprint arXiv:1603.08155.

Authorized licensed use limited to: National Taiwan University. Downloaded on August 20,2025 at 13:52:32 UTC from IEEE Xplore. Restrictions apply.

[6] Isola, P., Zhu, J. Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1125-1134).

[7] Beard, S. (2016). Using Fabric.js to Enhance Canvas Applications. Journal of Web Development, 10(4), 22-30.

[8] Zhu, J. Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In Proceedings of the IEEE international conference on computer vision (pp. 2223-2232).

[9] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[10] Huang, X., Liu, M. Y., Belongie, S., and Kautz, J. (2018). Multimodal unsupervised image-to-image translation. In Proceedings of the European conference on computer vision (ECCV) (pp. 172-189).

[11] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention (pp. 234-241). Springer, Cham.

[12] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., and Bengio, Y. (2014). Generative adversarial nets. Advances in neural information processing systems, 27.

[13] Kingma, D. P., and Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.

[14] Simonyan, K., and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[15] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

[16] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.

[17] Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3156-3164).

[18] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3431-3440).

[19] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 28, 91-99.

[20] Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2017). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence, 40(4), 834-848.

[21] He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).

[22] Huemint, [Online] Available: https://huemint.com/

Authorized licensed use limited to: National Taiwan University. Downloaded on August 20,2025 at 13:52:32 UTC from IEEE Xplore. Restrictions apply.