JavaScript 與 AI 技術在自動填色中的協同應用研究與優化

[1] 何承遠, [2] 龔郁雯, [3] 陳楷勳, [4] 李秀, [2] 龔郁婷, [5] 陳韋翰, [6] 陳薇安

[1] 國立臺灣大學 資訊管理系
E-mail: tommyho@ntu.edu.tw
國立臺灣大學 [2] 國際企業系, [3] 會計系, [4] 資訊工程系, [5] 經濟系, [6] 工商管理系
E-mail: {b09704048, b10612035, b09902082, b09704010, b12303134, b10701131}@ntu.edu.tw

摘要

本文描述一個結合 JavaScript 和 AI 技術的自動填色系統，旨在提升使用者體驗，推動教育創新，並提高創作效率。實作技術記錄使用 Canvas 元素技術和 Huemint 配色模型實現 AI 填色的具體步驟，包括載入圖層資料、排序區塊大小、監聽滑鼠/手指點擊、判斷使用工具及填色邏輯等。最後，本文列出不同情境下適合的配色模型，並比較各種技術優缺點。總結來說，本文展示如何在資源有限的情況下，協同 JavaScript 和 AI 技術實現高效率且靈活的自動填色解決方案。

關鍵字: JavaScript; AI 填色; 生成式對抗網路; Canvas; Huemint

1. 前言

隨著科技的進步與數位技術的普及，傳統繪本與著色書逐漸數位化轉型。在教育和娛樂領域，數位繪本和著色工具不僅提供更豐富的互動體驗，還能增強學習效果和激發創造力。現有的數位著色技術已經引入人工智慧(AI)自動填色功能，但在使用性、便利性、輔助程度和創意潛力方面仍有很大的提升空間。本文的開發動機主要來自於以下三個方面：

1. **增強使用者體驗**：現有的著色工具雖具備一定智能化功能，但其輔助能力仍有提升空間，讓使用者能更加便捷地操作，享受更豐富多彩的創作體驗。
2. **推動教育創新**：在教育領域，現有的 AI 填色技術在某些情境下仍存在著色不夠合理或不夠生動的問題。改進和優化 AI 自動填色技術能幫助學生更快速且準確地完成著色任務並吸收繪本內的知識。
3. **提升創作效率**：對於休閒時間零碎的青少年來說，手動著色耗時費力。AI 自動填色可以大幅提高創作效率，讓青少年能夠更有效率地運用休閒時間進行學習、玩樂與創作。

因此，本文將描述一個結合 JavaScript 和 AI 技術的自動填色繪本系統 [1]，為使用者提供智能化的著色體驗、推動數位繪本和著色工具的創新發展。另一方面，本文將描述開發過程中的技術選型、實作手段以及面臨的挑戰和其解決方案，期望能為相關領域的研究和實踐提供有價值的參考。

2. 現有 AI 自動填色技術回顧

2.1. 技術介紹

AI 自動填色技術近年來取得顯著進步，主要歸功於深度學習(Deep Learning, DL)和生成式對抗網路(Generative Adversarial Network, GAN)的發展。以下列出六個現有技術的狀態與優缺點：

1. **傳統圖像填色技術**：主要依賴手動操作，使用者需選擇顏色並逐步填充圖像區域。這些工具基於簡單的畫筆和填充工具，對於複雜圖像的填色需大量時間和精力，且缺乏智能輔助功能。
2. **基於規則的自動填色**：早期自動填色技術基於預定義規則，根據圖像中的線條和區域來確定填色模式。然而，這些技術的效果有限，無法靈活適應不同圖像的複雜性和多樣性。
3. **DL 技術應用**：隨著 DL 技術的興起，研究者開發了基於卷積神經網路(Convolutional Neural Network, CNN)的自動填色模型，能從大量標註數據中學習顏色模式，並自動將顏色應用到新的黑白圖像上，顯著提高填色效果的品質和自然度 [2, 3]。
4. **GANs**：一種以兩個 CNNs 組合而成的神經網路，透過一個 CNN 做為生成器(Generator)，另一個做為鑑別器(Discriminator)，讓 GANs 能生成高質量的彩色圖像，捕捉細緻的色彩變化和紋理，進而在圖像生成和填色方面表現出色 [2, 3]。
5. **半自動填色工具**：現有一些半自動填色工具結合手動操作和 AI 自動填色。使用者可以註記顏色標示，AI 模型根據標示自動補全，既保留使用者的創意空間，又能大幅提高填色效率和效果。
6. **商業軟體與開放原始碼項目**：已有多款商業軟體和開放原始碼項目實現 AI 自動填色功能。例如：1. 商業軟體 Adobe Photoshop 引入基於 AI 自動填色功

能，讓使用者可以快速為黑白照片添加顏色；2. 開放原始碼的 DeOldify、Image-Colorization 和 Image colorization with GANs 等利用 DL 技術實現高質量的圖像填色 [4]。這些開放原始碼工具在 GitHub 上獲得了廣泛關注和使用。

2.2. 技術限制與挑戰

儘管 AI 自動填色技術取得顯著進步，仍面臨至少如下的四項挑戰 [5-11]：

1. **顏色準確性**：自動填色模型有時無法準確捕捉圖像的真實顏色，可能產生不合理的色彩分佈。
2. **上下文理解**：現有模型在理解圖像的語義和上下文方面仍有不足，導致在複雜場景中表現不理想。
3. **訓練數據依賴**：模型性能高度依賴訓練數據的質量和多樣性，數據不足或偏向特定風格會影響填色效果。
4. **計算資源需求**：DL 模型的訓練和執行需巨量計算資源，對於資源有限的小型團隊和個人是挑戰。

因此，本文的目標是為資源有限的小型團隊和個人程式設計者找到突破上述挑戰的解決方案，期望進一步提升 AI 自動填色的準確性和智能性，為使用者提供更好的著色體驗。

3. 研究方法與實做技術

本文採用比較分析法，蒐羅現有 JavaScript 填色技術，對比各項技術的優勢與限制，進而針對本文的繪本系統網站開發客製化程式，供有製作著色網頁需求者參考。為了完整論述程式設計邏輯，本文將從如何實現著色效果切入，並講述其如何與 AI 填色方法結合。最後，提出本做法的優缺點、挑戰以及現有技術和本文做法各自適用的情境，期望未來研究者得以持續深化，使 AI 填色技術更臻成熟完備。

3.1. 思路簡介

為了定義繪本頁面的色塊分布、深淺差異，以及考量 AI 自動填色時的合理性，程式的第一要務是定義著色的「遊戲規則」。本方法是匯入必要的圖層，做為填色時的參考。這些圖層不會因為使用者的操作而改變，一切結果只呈現在給使用者的畫布上。以下將詳細介紹各圖層的功能與效果：

1. **框線圖層**：框出不同色塊的邊界，並包含無需上色區塊的顏色，而可供上色的區域在本圖層中為透明。框線圖層直接呈現給使用者，並覆蓋於使用者的畫布之上，如圖 1 所示。
2. **色塊圖層**：定義為應被著上一致顏色的區塊。該圖層不包含框線，透明的部分表示無法上色，有顏色的部分則根據顏色劃分畛域。如圖 2，以「人」為例，若在色塊圖層中將此人的頭髮設定為藍色，

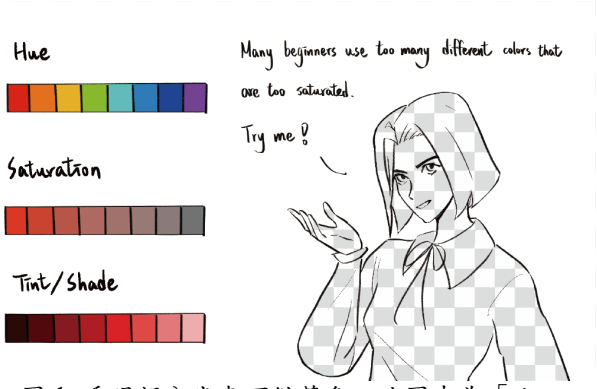

圖 1: 透明部分代表可供著色，此圖中為「人」

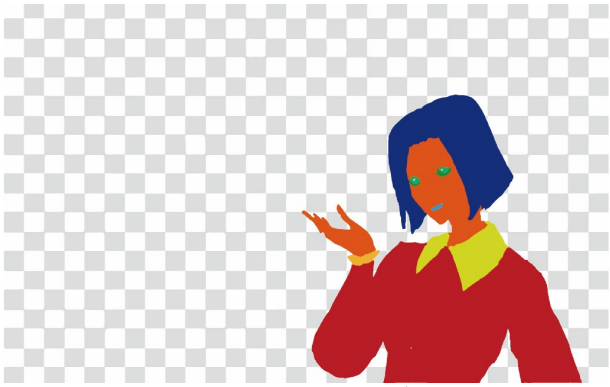
圖 2: 為不同顏色的區塊定義及其範圍

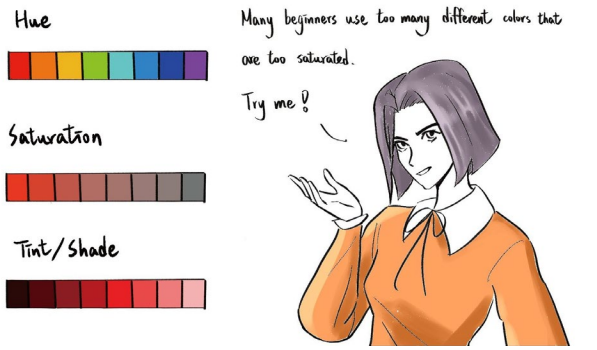

圖 3: 透過色塊圖層實現頭髮和衣服為不同顏色

將衣服設定為紅色，則當使用者點擊頭髮時，所有藍色的部分在呈現給使用者的畫布上會換上所選擇的畫筆色彩，正好是頭髮的範圍。此方式可以設定不同區域為不同區塊，用以填上不同的顏色，如圖 3 中的頭髮和衣服為不同區塊。
3. **灰階圖層**：界定顏色的深淺，為繪本草擬深淺效果，故無透明部分，越接近白色表示顏色越明亮，反之亦然。再以「人」為例，如圖 4 和圖 5，即使衣服在色塊圖層中為單一顏色，在疊上灰階圖層並經過調深淺後，渲染在灰階圖層中對應的明暗位置，使衣服看起來具有反射和陰影。
4. **固定色彩圖層**：只適用於 AI 自動填色情境，如前所述，本文系統的宗旨為寓教於樂，有些區域的顏色有標準答案，如鋅燃燒時會產生黃綠色火焰。此時，使用者必須將火焰的顏色塗成黃綠色方為正

解；又有些區域的顏色須具備一定合理性，像是人的皮膚會接近橘白色或咖啡色。因若採取無限制的 AI 自動填色，搭配出來的顏色固然賞心，卻不必然正確和合理，這將失去教育意義，如同圖 6 和圖 7 所示。另外，固定色彩圖層中透明的部分表示並未指定顏色，而有顏色的部分則直接將該顏色賦在呈現給使用者的畫布上的相應位置，確保使用者看到的配色合情合理。

圖 4: 界定同一區塊內的顏色深淺變化

圖 5: 左右圖分別為不使用與使用灰階圖層的結果。相比之下，可看出右方顏色更有層次

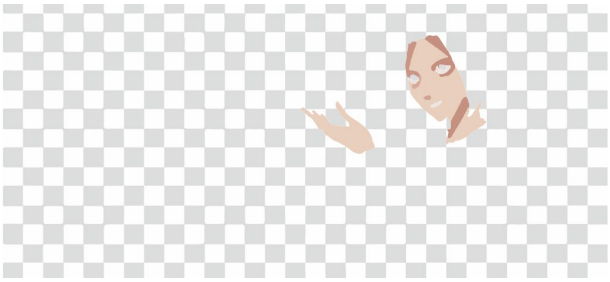

圖 6: 為 AI 填色提供合理的答案

圖 7: 左右圖分別為不使用與使用固定色彩圖層的結果。相比之下，可看出右方皮膚顏色合理許多

3.2. 具體實現步驟

以下描述填色系統具體實現步驟：

1. **載入圖層資料**：在頁面載入時，系統會先行載入框線圖層、色塊圖層、灰階圖層、固定色彩圖層及編輯成果圖層，並使用 Canvas 元素提取這些圖層的所有像素之三原色(RGB)。這些圖層不會改變，其作用是供上色邏輯參考。前端(網頁)和後端(MySQL)的互動過程如圖 8 所示。先將圖層群的 URL 由 http 網址轉成 base64 格式，再將 URL 綁在對應的指令上，如 image.src。此轉換步驟實屬必要，雖然 base64 格式的 URL 能避免 CORS 錯誤，但 base64 格式資料量極大(8 萬個字元)，因此實作上以 http 形式存在資料庫，並在要將圖層群傳至前端時處理。

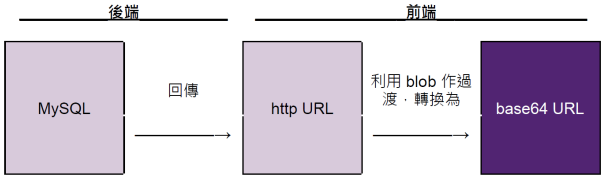

圖 8: 圖層 URL 轉換過程，最終為 base64 格式

2. **排序區塊大小**：依據色塊圖層中的記錄，系統會在內部進行區塊大小排序，保證上色時不會因為順序問題導致色塊顏色覆蓋。本系統實作上採用的配色 AI 模型為 Huemint，以 12 個色碼為基礎搭配，如圖 9 為一示意圖。

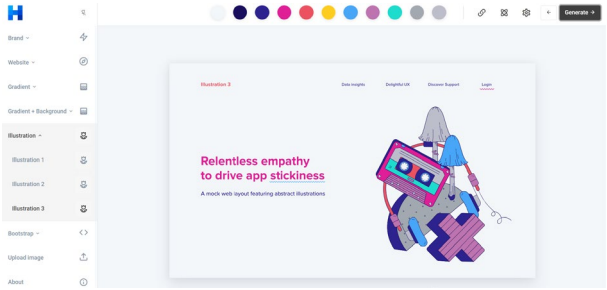

圖 9: 示意圖，藍白色適用於背景，為第一個顏色

3. **監聽滑鼠/手指點擊**：使用者在畫布上點擊時，系統會根據點擊位置，參考色塊圖層的顏色資訊，決定顏色填充區域。如圖 10 所示，先計算點擊位置的 x/y 值與畫布左/上邊界的差距，以及該差距占畫布寬度/高度的比例。再將該比例乘上畫布寬度/高度的單位，得出單位的 x'/y'，並以 y'乘以畫布寬加上 x'找出目標像素，得知其在色塊圖層中的顏色。

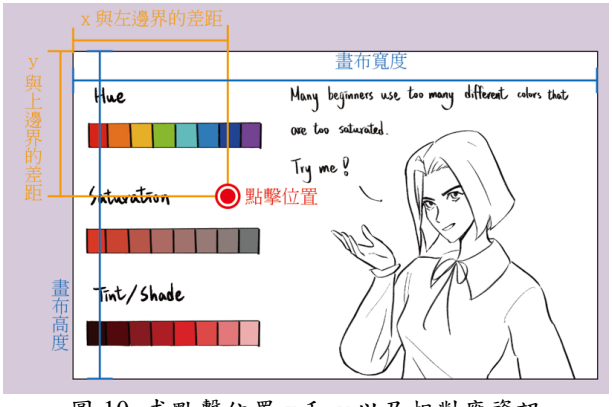

圖 10: 求點擊位置 x 和 y 以及相對應資訊

4. **判斷使用工具**：根據使用者選擇的填色工具，如畫筆、橡皮擦、AI 魔術棒、重設畫布，同時進一步決定是使用單色填充、漸變填充或圖案填充。
5. **上色邏輯**：系統根據判斷的工具及色塊排序圖層進行顏色填充或清除，確保顏色不會錯誤覆蓋。

6. **儲存圖片**：當使用者結束編輯，系統會將最終成果轉換成 png 格式的圖檔，上傳至 Bytescale 圖床空間，並將上傳後產生的 http 格式 URL 回傳至後端儲存，如圖 11。另外，為了圖床的儲存空間不要被之前成果佔據，故上傳新成果前會先將舊檔案刪除。

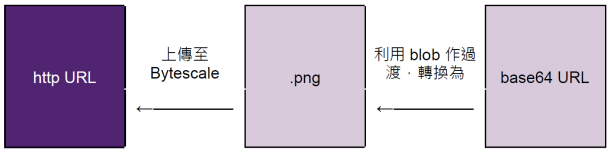

圖 11: 圖片 URL 轉換過程，最終為 http 格式

3.3. AI 填色技術結合方法

此小節進一步說明本系統與 AI 填色技術結合方法 [12-21]。

3.3.1. 配色模型選型

在考量配色邏輯時，本系統結合 Huemint 配色模型。Huemint [5, 22]是一款基於 GANs 的自動配色工具，能根據輸入的色彩方案生成多樣且協調的配色方案。以下介紹其應用步驟：

1. **匯入 Huemint 模型**：在頁面載入時，系統會匯入 Huemint 模型，用於生成初始配色方案。
2. **生成初始配色**：系統會在使用者首次點擊畫布時，根據 Huemint 模型生成初始配色方案，如圖 9 所示，這些配色會保存在色塊圖層中，供後續填色參考。
3. **動態配色生成**：當使用者選擇不同區域或工具時，系統會動態生成新的配色方案，確保顏色協調且多樣。

3.3.2. 結合上色邏輯

本系統將 Huemint 模型生成的配色方案與前述填色邏輯結合，確保顏色填充的準確性和美觀性。以下描述結合步驟：

1. **配色方案應用**：在判斷填色區域時，系統會根據 Huemint 模型生成的配色方案決定填色顏色。
2. **動態調整**：根據使用者的操作，系統會動態調整配色方案，確保顏色填充的和諧性。
3. **優化填色效果**：系統會根據色塊排序圖層的資訊，優化填色效果，避免顏色覆蓋錯誤。

4. 研究方法討論

由前述的描述可知本文並非針對 AI 模型本身進行優化，而是透過制定遊戲規則，借助演算法進行優化和達成合宜的結果，故適合初學者理解和學習之用；也適合想要實現 AI 填色功能，但計算資源有限的小型團隊和個人，並在不用自行訓練 AI 模型的狀況下開發。接下來，將討論本方法與其他填色技術，如可與 AI 配色 API 結合的 JavaScript 填色技術和現有 AI 填色技術 [1-22]。

4.1. 與其他 JavaScript 填色技術比較

在 JavaScript 填色技術中，現有 JavaScript 填色技術可以分成兩大類來探討。

1. **在可填色區域下方放置 HTML 元素(簡稱 HTML 元素換色)**：這種方法將圖片上待填色區塊挖成透明，並在每個區塊的相應位置放置一個 HTML 元素，如 canvas、SVG、fabric.polygon 或 div。當使用者點擊可填色區域時，底下的 HTML 元素會被填入選定的顏色。優點是程式簡潔，易維護，但難以處理不規則形狀的區域，且無法產生自然的陰影和反光效果。
2. **將數個可填色區域分別存成圖片，並賦予不同的 Alpha 值(簡稱可填色區塊不同 Alpha 值)**：此方法將所有不規則的填色區域存成 png 圖片，並為每個區域的像素點賦予不同的 Alpha 值。當使用者點擊可填色區域時，與點擊位置擁有相同 Alpha 值的像素點會被填入選定的顏色。此方法便於製圖者存取圖層圖片，但難以定位圖層位置，程式缺乏彈性，且匯入圖片時間較長。

而本文章技術通過將圖片中的填色區域轉換為色塊圖層，利用 AI 模型進行配色，並自動填色。此方法能處理形狀複雜且填色區域多的圖片，具有高度的精準度和自然的上色效果，但製作圖層較為費工。

上述三種填色技術的特點、優缺點和其適用情境如表 1 所示。

4.2. 與現有 AI 填色技術比較

比較訓練 GANs 的直接方法與本文章使用配色 AI 模型的間接方法，主要區別在於執行難易度、上色速度和顏色精準度。

1. **直接訓練 GANs**：需大量數據餵養 AI 模型，訓練過程耗時耗力。上色速度視模型而定，約需 3-8 秒。顏色精準度取決於模型訓練的品質，但對於非寫實風格的圖片，精準度較低。
2. **間接使用配色 AI 模型**：本文章技術利用 Huemint API 取得一組配色，並依序填入顏色。此方法易於實施，約需 8 秒回傳配色結果。透過固定色彩圖層指定顏色的部分精準度高，其他部分則較低。

上述兩種 AI 填色技術的特點與適合情境等項目如表 2 之比較。

由表 1 和表 2 可得知，本文章技術適合處理形狀複雜、填色區域多且相鄰的圖片，並能在保留著色彈性的同時，產生自然的陰影和反光效果。

表 1: 三種 JavaScript 填色技術比較表

技術	HTML 元素換色	可填色區塊不同 ALPHA 值	本文章技術
符合區域邊界	難	易	易
灌色難易度	易	中	難
圖層匯入時間	無圖層	不固定	較固定
上色層次	無層次或層次不自然	可以有陰影或反光效果	可以有陰影或反光效果
優點	程式簡潔、易維護	便於製圖者存取圖層圖片	處理複雜且填色區域多圖片
缺點	無法隨區域形狀賦形	難以定位圖層位置，缺乏彈性	製作圖層費工
適用情境	填色區域大且分散	填色區域較少	填色區域多且相鄰

表 2: 兩種 AI 填色技術比較表

技術	直接訓練 GANs	本文章技術(間接使用配色 AI 模型)
執行難易度	難，培養模型耗時耗力	易
上色速度	約 3-8 秒	約 8 秒
顏色精準度	中至低	視區域指定顏色與否而定，高至低
可用顏色數	不限	至多 12 個顏色
模型依賴	高	中
適用情境	・專案團隊資源豐富 ・圖案需填色的區塊較多 ・可以找到大量相同風格的資料訓練模型	・專案團隊小或為個人 ・圖案需填色的區塊較少 ・圖案具藝術家個人特色，現有資料少

5. 結論

通過結合 JavaScript 和 AI 技術，本文實現一個高效能、高品質且富有智能的自動填色系統。本系統不僅提升使用者體驗，還推動教育創新和創作效率的提高。未來將針對「AI 模型提供的配色數限制」、「相鄰色塊交界之混色問題」和「根據情境尋找最適顏色」等，進一步優化本系統，提升 AI 模型的準確性和智能性，並探索更多應用場景和技術結合的可能性，為使用者提供更好的著色體驗。

參考文獻

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

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

[3] Krizhevsky, A., Sutskever, I., & 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., & Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. arXiv preprint arXiv:1603.08155.

[6] Isola, P., Zhu, J. Y., Zhou, T., & 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., & 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., & 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., & 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., & 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., ... & Bengio, Y. (2014). Generative adversarial nets. Advances in neural information processing systems, 27.

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

[14] Simonyan, K., & 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., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

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

[17] Vinyals, O., Toshev, A., Bengio, S., & 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., & 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., & 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., & 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., & 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/