Integration of React.js with U-Net Architecture for Image Segmentation Abhinay Goswami, Divakar Sharma, Jay Kishor Thakur

by Mamta Rajput

Submission date: 25-Mar-2025 12:14PM (UTC+0530)
Submission ID: 2624639243
File name: Integration_paper.pdf (256.78K)
Word count: 3188
Character count: 19752

Integration of React.js with U-Net Architecture for Image Segmentation

Abhinay Goswami, Divakar Sharma, Jay Kishor Thakur
Department of Artificial Intelligence & Data Science, Galgotias College of Engineering & Technology Greater Noida Address
abhinay.21gcebaids059; divakar.21gcebaids009; jay.21gcebaids026@galgotiacollege.edu

Abstract **-** This paper highlights the integration of a Deep Learning Model and client - side interface , which is a global transformation in shaping the future of modern web applications related to emerging trends in machine learning and Artificial Intelligence.This research explores how React.js, a popular front-end javascript library as well as framework, can be integrated with a U-Net based segmentation model to create a visually appealing and efficient web-based interface for real-time image processing.This paper enhances the growing field of AI-enabled web applications by providing insights into the deployment, scalability, and real-time processing. Through this study, we highlight both the possibilities and challenges of integrating React.js with deep learning models.

Keywords- Image Segmentation; U-Net; React.js; Deep Learning; Web - AI Merger; Real-Time Image Processing; Full stack Development

I.Introduction

1.1 *The Significance of Image Segmentation in Computer Vision* : Image segmentation plays a significant role in **computer vision** that includes breaking an image into multiple meaningful subgroups known as **image segments**. Unlike traditional **image classification**, where an entire image is assigned a single label, segmentation provides a technique to assign labels to pixels, which reduces the complexity of analysing image and precise localization.Segmentation follows two approaches , which are based on similarity and discontinuity.Similarity approach detects the similar image pixels , whereas Discontinuity approach works on the discontinuous value of image pixel intensity.There are several types of Image Segmentation : Semantic segmentation, Instance segmentation ,Panoptic segmentation etc. Image finds the application in areas of various domains :

A. *Medical Imaging* : Segmentation is essential for the analysis of medical scans such as X-rays, MRIs, CT scans, and ultrasound images. With the integration of advanced techniques in image segmentation , Tumor Detection ,Organ Segmentation: Retinal Image Analysis,Pathology has become an achievable field in the modern era.

B. *Remote Sensing & Satellite Image Analysis* : Satellite imagery provides ample amounts of data for analyzing geo-physical changes. Image segmentation is used for Land-driven classification, Climate monitoring ,and Disaster management.

C. *Autonomous Vehicles & Traffic Monitoring* : Self-driving cars depend on segmentation for environmental response and decision-making. Computer vision models segment objects in a scene to recognize:Road patterns and boundaries , Vehicles and obstacles , Pedestrians and cyclists , Traffic signs and signals.

D. *Industrial Automation & Quality Control* : Factories automate computer vision models to configure products during manufacturing. Segmentation helps in Defect detection , Sort and count process and Package verification.
Deep learning, particularly CNN-based architectures, has significant improved image segmentation accuracy and efficiency. Traditional segmentation methods, such as : Thresholding , Edge detection , Region-based segmentation have limitations in handling large datasets, occlusions, and varying lighting conditions.

1.2 *U-Net: A Discovery in Deep Learning for Image Segmentation* : Deep learning has essentially advanced the field of image segmentation, model to achieve high accuracy in detecting objects at an optimum pixel level. One of the most influential architectures for this task is U-Net, introduced by Olaf Ronneberger et al. (2015). U-Net follows a Fully Convolutional Network (FCN) structure, comprising two main components:

A. *The Encoder (Contraction Path)* : The encoder is solely responsible for extracting spatial features from the inputted image. It comprises multiple convolutional layers followed by activation functions and max pooling to gradually reduce specific dimensions.This decreasing sampling process enables the model to capture extreme-level representations of objects in the image.

B. *The Decoder (Expansion Path)* : The decoder reconstructs the image from its contracted form, restoring spatial resolution and evoluting the segmentation mask. It uses incremental sampling layers and transpose convolutions to gradually enlarge feature maps.This ensures that the final output preserves the original image dimensions while highlighting key segmented areas.

Skip Connections : A dedicated feature of U-Net is its skip connections, which directly propagate featured maps from the encoder to the decoder at extreme levels.These connections help in retaining fine details and spatial information, preventing the loss of important features during decreased sampling.
The key reasons why U-Net is widely used in segmentation tasks are:

A. *High Accuracy with Limited Data* : Unlike many deep learning models that require large amounts of labeled data, U-Net is suitable with mini datasets.

B. *Preserves Spatial Information* : The use of skip connections maintains the important structural details, improving segmentation accuracy for complex images with optimum details.

C. *Versatile Across Various Image Types* : U-Net works with grayscale, RGB, and multi-ranged medical images, making it applicable across various domains such as medical research, agriculture, and industrial automation.

D. *Computational Efficiency* : Despite its deep architecture, U-Net is relatively efficient compared to more complex models, enabling real-time segmentation in applications that demand fast retrieval.

Despite its advantages, integrating U-Net into real-time modern web applications causes various challenges that must be observed to ensure fast and seamless deployment.Web applications must handle large image files, requiring appropriate data transmission between the frontend and backend.These challenges underlines the importance of choosing an appropriate frontend framework that can seamlessly integrate with U-Net to deliver real-time segmentation results.

1.3 React.js: Revolutionizing Frontend Development for Deep Learning Models : Modern web applications demand interactive , visually appealing and efficient user interfaces to enhance user experience and performance. React.js, developed by Meta , has emerged as the most powerful JavaScript library as well as framework for building dynamic and responsive modern web applications. Frontend plays a crucial role in handling user interactions and ensuring a smooth flow of data between the client and server. React.js offers various advantages to AI-based applications, particularly that require real-time processing , state management, and appropriate API communication. The following features make React.js a perfect choice for integrating deep learning models into web applications:

A. *Reusable Component Architecture* : React follows an efficient model where applications are built using independent and reusable components.

B. *Virtual DOM* : React's Virtual DOM optimises performance by minimizing unnecessary re-renders, ensuring smooth UI updation.

C. *State Management* : AI-based applications often involve real-time data updates, requiring efficient state management.React having built-in Hooks (useState, useRef , useCallback , useEffect) and external libraries like Redux and Context API to handle dynamic UI changes appropriately.

D. *API Integration with AI-based Models* : React.js enables smooth communication with backend AI-based models through REST APIs and WebSockets.It can send image input data to the backend for analysis and receive masked - segmentation results asynchronously.

1.4 FastAPI : A highly efficient backend : FastAPI is a modern backend framework designed for highly efficient, and asynchronous API development. Built on Starlette and Pydantic python libraries, it includes automatic request validation, and modern real-time data processing abilities, making it the perfect choice for a U-Net deep learning model like U-Net. Some of the main highlights of FastAPI are:

A.*Asynchronous API development with ASGI Support* : Unlike old synchronous python frameworks like Django and Flask , FastAPI is the only one that supports ASGI (Asynchronous Server Gateway Interface), enhancing unblocking request handling. This allows multiple segmentation tasks to run parallelly, essentially improving response.

B. *High-Performance Process* : FastAPI is made on Starlette,popularising it as one of the efficient Python web frameworks available. Benchmark tests show that FastAPI can handle thousands of requests per second, outsmarting Flask by three times in speed.

C. *Data Validation* : FastAPI includes Pydantic to verify request data structures. This makes sure that only efficiently formatted image files are executed by U-Net, protecting system corruption.

D. *WebSockets for Real-Time Data Streaming* : Unlike REST APIs, which work on polling for updates, WebSockets in FastAPI stabilise real-time streaming of segmentation results. This is compulsory for interactive and efficient AI applications, where users need immediate feedback on their inputted images.

FastAPI connects quickly with PyTorch, CUDA GPUs and TensorFlow, permitting the U-Net model to perform model integration efficiently. Since deep learning models demand high computational power, running them on GPU-accelerated instances can reduce inference time by ten times compared to CPU execution. FastAPI's capability to process requests asynchronously makes sure that the backend does not become useless when handling multiple segmentation tasks concurrently. Its support for concurrent processing using task queues like Celery. This allows the system to distribute multiple segmentation tasks efficiently across CPU/GPU resources.

This research paper explores the efficient integration of React.js with U-Net for real-time image segmentation using FastAPI as the backend framework. By employing FastAPI's asynchronous processing capabilities, this research aims to develop an appropriate , visual-appealing and interactive web-based modern application that allows users to upload images and process them through U-Net.

II.THEORETICAL FOUNDATION

The proposed system follows a client-server model, where the frontend UI(React.js) integrates with the backend (FastAPI) that hosts the deep learning model (U-Net model). The backend

of our system is responsible for U-Net model analysis, image processing and API management, while the frontend provides an interactive and responsive user interface.

2.1. Frontend Functionality in Integration : The frontend made using React.js is efficiently responsible for handling user interactions, processing API (Application Programmable Interface) communication, and dynamically rendering segmentation masks.Here is the workflow of frontend for integration system:

A. *Image uploading* : The UI using react.js allows users to select an image from their device and preview it before submission using FileReader API.

B. *Image sent to Backend* : React.js sends the user - uploaded image to the backend via Axios (REST API) or WebSockets and FormData API ensures the image is transmitted correctly in the request payload.

C. *FastAPI Processes the Image* : The backend validates, resizes, and normalizes the user-inputted image before passing it to the U-Net model. The front end shows a loading indicator while processing occurs over the backend.

D. *U-Net Generates a Segmentation Mask* : The U-Net model performs pixel-wise image segmentation and produces an output mask. The front end listens for updates using WebSockets or a promise-based API response.

E. *Output to user* : The front end displays the Segmentation Output received from backend.The segmentation mask is dynamically converted onto the desired image.Users can adjust opacity, compare results or download the segmented image.

2.2 U-Net's Integration with FastAPI : The U-Net model is a Fully Convolutional Neural Network (FCN) of deep learning designed for pixel-based segmentation tasks. It follows an encoder-decoder structure with the skip connections for linking encoding and decoding (explained in introduction section in detail).Here is the step flow for processing:

A. *Preprocessing input images* : User - inputted Images are validated , resized and normalized using OpenCV and NumPy.Tensor transformation (PyTorch/TensorFlow) ensures model compatibility of the inputted image.

B. *Model inference* : The pre-processed image generated after normalization is passed through the U-Net model, which predicts the segmentation mask. GPU acceleration (CUDA, TensorRT) is applied for faster and efficient processing.

C. *Post-processing output masks* : The output is converted into a human displayable image format (PNG, JPEG).Thresholding techniques may be applied to enhance the resolution of particular features and refine results.

D. *Returning results to the frontend* : The segmented image is sent back to the frontend via REST API or WebSockets. The result is stored temporarily in Local Storage for comparison.

2.3 FastAPI : Backend Processing Pipeline **:** The backend processing pipeline in FastAPI serves as the interface between the frontend and the U-Net deep learning model. It is responsible for efficiently handling image upload requests, preprocessing the input, running AI model inference, post-processing segmentation outputs, and sending results back to the frontend for visualization.

A. *Receiving image upload requests* : When a user uploads an image via the frontend, FastAPI processes the request using REST APIs (Axios) or WebSockets (socket.io). The image is validated with Pydantic (Python library for validation), ensuring correct format, size, and resolution. Pillow (PIL) checks image metadata, while Uvicorn ASGI server optimises high-speed handling.

B. *Preprocessing images* : Validated images are preprocessed using OpenCV and NumPy to ensure compatibility with the U-Net deep learning model. Images are resized, normalized, and converted into numerical arrays before being transformed into tensors via TorchVision (for PyTorch) or TensorFlow Image Processing.

C. *Running model inference* : The processed image is fed into U-Net, which generates a pixel-based segmentation mask. Inference is ascended using CUDA (for GPU processing), TorchScript or ONNX. To prevent API blocking, FastAPI Background Tasks can handle inference asynchronously.

D. *Post-processing the segmentation mask* : The U-Net output is processed into a binary segmentation mask using NumPy and OpenCV. If necessary, Matplotlib applies color mapping, and Pillow (PIL) converts the output into PNG/JPEG format.

E. *Returning results to the frontend* : The processed segmentation mask is returned to the frontend using either a REST API response (JSON with Base64 encoding) or WebSockets for real-time streaming. The frontend fetches the image and overlays it on the original using React Canvas API (Konva.js).

Despite its efficiency, the integration of React.js, FastAPI and U-Net poses challenges such as handling large image datasets, reducing model inference latency, optimizing frontend-backend communication, and ensuring scalability. These issues and their potential solutions will be explored in the next section.

III. CHALLENGES IN INTEGRATION

The integration of React.js (frontend), FastAPI (backend), and U-Net (deep learning model) introduces various technical and interfacing challenges that must be addressed to ensure smooth communication and essential data processing.

A. *Discontinuity in Frontend-Backend Communication* : React.js executes asynchronously, while FastAPI follows a input-response model by default, leading to delays in getting segmentation results. Since REST APIs depend on polling, React.js must iteratively request updates, increasing server load and affecting real-time performance. This latency in execution flow creates latency issues.The process of sending inputted images, processing requests, and returning results causes noticeable lags.Repeated API calls from React.js strain the backend, reducing FastAPI's efficiency. FastAPI processes requests synchronously, whereas React.js includes real-time updates, creating timing latency.

B. *Handling Large Image inputs efficiently* : Inputting high-resolution images from React.js to FastAPI includes bandwidth constraints, memory storage issues, and processing delays. Since image segmentation receives pixel-based accuracy, compressed images may be several megabytes in size, increasing data transmission time. When multiple users upload input images simultaneously, server performance declines, leading to slow output times.Large image sizes increase transmission time, delaying segmentation.Storing multiple images in memory causes backend bottlenecks, slowing inference.Image files require conversion before they can be processed, adding extra computation time.

C. *Enhancing Real-Time U-Net Model Inference* : The U-Net model is sequentially intensive, requiring important CPU/GPU resources for efficient segmentation. Running deep learning interfacing on the backend introduces discontinuity , increases execution time, and can overwhelm server resources, especially under high concurrent request loads.Processing a single image on CPU-only environments takes various seconds, protecting real-time segmentation.Deploying U-Net on GPUs improves speed, but GPU resources must be optimized for multiple users to prevent performance drops.

D. *Accurately transmitting Segmentation results to the frontend* : When U-Net returns a segmentation mask, FastAPI must transmit the outputted image back to React.js. However, network bandwidth constraints and mismatched data handling can increase API response times, causing delays in getting segmentation results.Getting large segmentation masks slows API responses.Conversion of model output into image various formats adds latency before sending results to React.js.

Rendering segmentation masks in React.js impacts UI responsiveness, leading to slow updates and frame drops.Scaling the system for multiple users requires efficient resource allocation, and cloud deployment increases costs and security risks, necessitating optimized DevOps strategies.

IV. CONCLUSION

The integration of React.js, FastAPI, and U-Net in real-time image segmentation discuss the potential of deep learning in modern web-based applications. This research enhances the frontend-backend interaction, the deployment of U-Net for segmentation, and the challenges in ensuring real-time processing. Key critical factors, including latency in API communication, handling large image files, inference delays, and UI performance limitations, were identified as critical challenges affecting system efficiency. While FastAPI enables high-performance API communication, and React.js provides interactive visualization, optimizing model inference, data transfer, and scalability remains essential.

V. FUTURE SCOPE

There are various areas where ahead research and development are needed to fully harness the potential of AI and machine learning in front-end development. One key area is refining AI algorithms to improve efficiency and reduce the computational resources required, making them more accessible for a wider range of applications and devices. The future scope of integrating React.js, FastAPI, and U-Net lies in enhancing performance, scalability, and real-time efficiency. Implementing WebAssembly (WASM) and TensorFlow.js can enable in-browser AI inference, reducing latency and server dependency. Enhancing frontend-backend communication with GraphQL and WebSockets can further streamline data transfer. Additionally, cloud-native solutions like AWS Lambda and Kubernetes can ensure scalability under high traffic.

References

[1] *M. M. Islam, D. Zhang and G. Lu. "A geometric method to compute directionality features for texture images", In Proc. ICME, (2008), pp. 1521-1524.*

[2] Benbya, H., Davenport, T. H., & Pachidi, S. (2020). Artificial intelligence in organizations: Current state and future opportunities. MIS Quarterly Executive, 19(4).

[3] Y. Sasaki, M. Fukui, J. Hagikura, J. Moriyama, and T. Hirashima, "Development of an interactive educational tool to experience machine learning with image classification," in Proc. IEEE 9th Global Conf. Electron. (GCCE), Oct. 2020, pp. 78–80.

[4] V. Johnston, M. Black, J. Wallace, M. Mulvenna, and R. Bond, "A framework for the development of a dynamic adaptive intelligent user interface to enhance the user experience," in Proc. 31st Eur. Conf. Cognit. Ergonom., Sep. 2019, pp. 32–35.

[5] Goh, H.-A., Ho, C.-K., & Abas, F. S. (2023). Front-end deep learning web apps development and deployment: a review. Applied Intelligence, 53(12), 15923-15945.

[6] *Kavya R and Harisha 2015 Feature Extraction Technique for Robust and Fast Visual Tracking: A Typical Review (International Journal of Emerging Engineering Research and Technology Vol 3 Issue 1) PP 98-104.*

[7] Deepika Jaswal, Sowmya.V and K.P.Soman, "Image Classification Using Convolutional Neural Networks", International Journal of Scientific and Engineering Research, ISSN 2229-5518, Volume 5, Issue 6, June-2014.

[8] Neha Sharma, Vibhor Jain and Anju Mishra, "An Analysis Of Convolutional Neural Networks For Image Classification",International Conference on Computational Intelligence and Data Science (ICCIDS 2018).

[9] *Jimenez-Munoz, Juan C.; Sobrino, Jose A.; Skokovic, Drazen et al. 2014. Land Surface Temperature Retrieval Methods from Landsat-8 Thermal Infrared Sensor Data. IEEE GEOSCIENCE AND RS LETTERS 11(10): 1840-1843.*

[10] *Pope, Allen; Rees, W. Gareth; Fox, Adrian J. et al. 2014. Open Access Data in Polar and Cryospheric Remote Sensing. Remote Sensing 6(7): 6183-6220.*

[11] *A. H. Stroud 1971 Approximate Calculation of Multiple Integrals(Prentice-Hall Inc., Englewood Cli_s, N. J.).*

[12] *Christopher Clapham, James Nicholson 2009 Oxford Concise Dictionary of Mathematics(OUP oxford).*

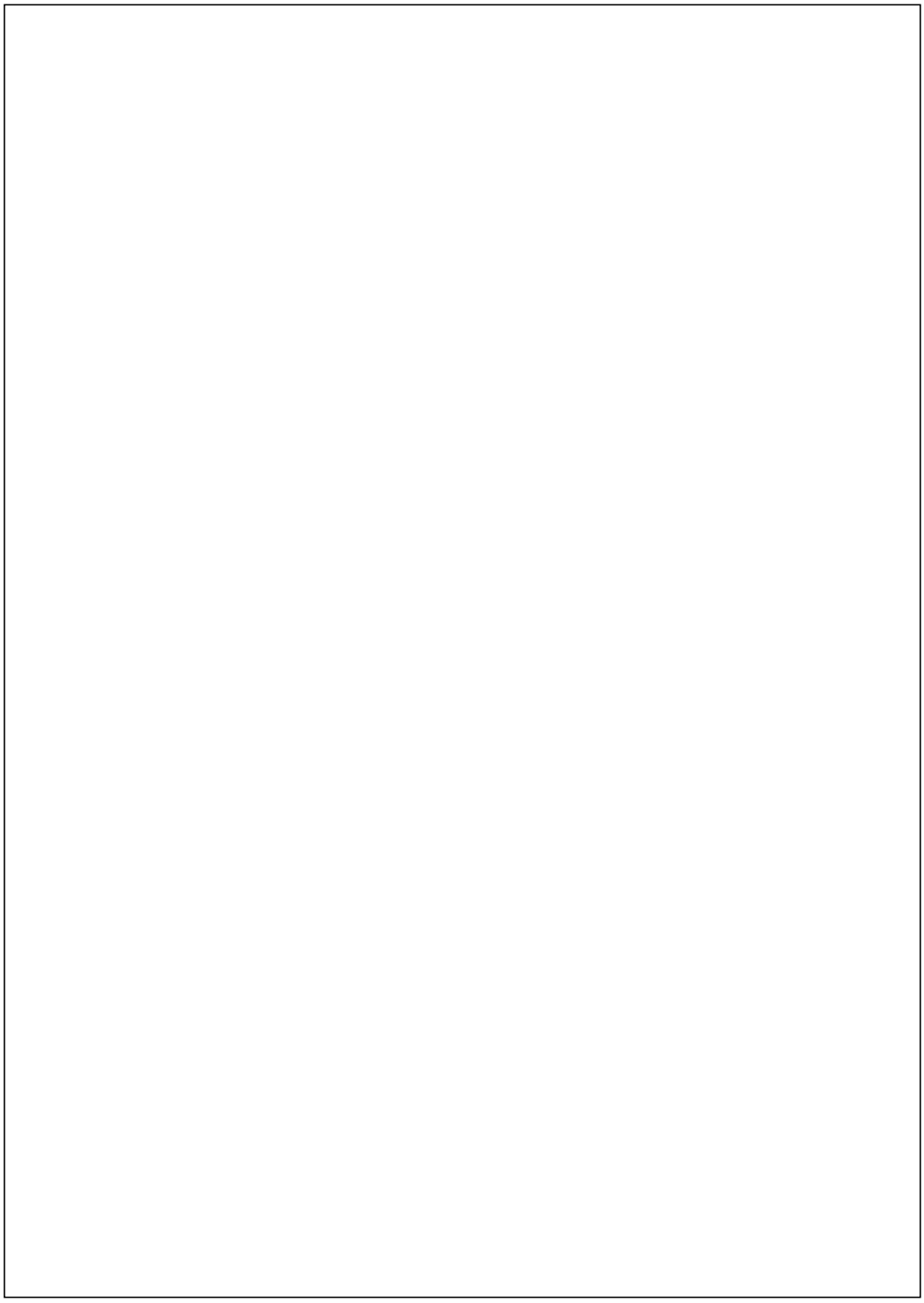
[13] *S. A. Patil and V. R. Udpi 2010 Chest x-ray features extraction for lung cancer classification(Journal of Scientific and Industrial Research Vol 69) pp 271-277.*

[14] *R. C. Gozalez and R. E. Woods 2002 Digital Image Processing Using Matlab, 2nd ed, Gatesmark (USA) chapter 12 pp 642-654.*

[15] *K. P. Aarthy and U. S. Ragupathy 2012 Detection of lung nodule using multiscale wavelets and support vector machine(International Journal of Soft Computing and Engineering (IJSCE) Vol 2, Issue 3).*

[16] *A. K. Jain and A. Vailaya, "Image retrieval using colour and shape", Pattern Recognition, vol. 29, no. 8, (1996), pp. 1233-1244.*

[17] *M. Flickner, H. Sawhney, W. Niblack, et al., "Query by image and video content: the QBIC system", IEEE Computer, vol. 28, no. 9, (1995), pp. 23-32*



Integration of React.js with U-Net Architecture for Image Segmentation Abhinay Goswami, Divakar Sharma, Jay Kishor Thakur

ORIGINALITY REPORT

4%	3%	0%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	gsjournals.com Internet Source	2%
2	www.geeksforgeeks.org Internet Source	1%
3	Arvind Dagur, Karan Singh, Pawan Singh Mehra, Dhirendra Kumar Shukla. "Artificial Intelligence, Blockchain, Computing and Security", CRC Press, 2023 Publication	<1%
4	Submitted to Universiti Teknologi Malaysia Student Paper	<1%
5	www.mdpi.com Internet Source	<1%
6	aiforsocialgood.ca Internet Source	<1%
7	dokumen.pub Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		