Enhancing Research Paper Summarization Through Advanced Language Techniques: Integrating Abstractive Methods, Fine-tuning Large Language Models, and Retrieval-Augmented Generation Technology

Indrajeet Roy

Abstract

The aim of this paper was to develop a novel system for summarizing academic research papers, leveraging the latest advancements in artificial intelligence and natural language processing. The methodology combined abstractive text summarization, which paraphrases key concepts using an encoder-decoder neural network with attention mechanisms, but it was learned that with a small amount of data, this technique did not produce good results. Therefore, two techniques were leveraged: Retrieval-Augmented Generation (RAG) technology along with neo4j to construct knowledge graphs and efficiently answer questions about the research paper, integrated with Large Language Models (LLMs). Secondly, the MistralAI with 7 billion parameters was fine-tuned for efficient text summarization. This integrated approach aspired to transform how researchers interact with vast volumes of academic literature, by providing concise, insightful, and context-rich summaries. The project addressed critical challenges, including data quality and availability, the complexity of abstractive summarization, and the computational demands of LLM fine-tuning, aiming to significantly enhance the research paper reading experience.

1 Introduction

Motivation: The motivation behind the research project stems from the growing need for reliable and accessible summarization of research papers. Existing applications offer summaries but often lack precision, and while solutions like chat GPTs provide more accurate summaries, they face limitations such as token count restrictions and come at a cost. The goal was to address these challenges by offering free, efficient summarization and interaction with research papers.

Project Overview: To achieve fine-tuning for research paper summarization, various Large Language Models (LLMs), such as LLaMA 2, were considered. Ultimately, Mistral AI, with its 7 billion parameters, was chosen due to its superior performance. Although this choice presents computational challenges, techniques like Parameter-Efficient Fine-Tuning (PEFT) and Low-Rank Adaptation (LoRA) were utilized to minimize these complexities. Additionally, the project leveraged Retrieval-Augmented Generation (RAG) for its ingenious yet straightforward methodology, enabling the implementation of a 'chat with PDF' feature. For a detailed explanation of these methods, please refer to the Methods section of this report.

2 Methods

2.1 Fine-tuning LLM

Large Language Models (LLMs) have revolutionized the field of natural language processing by offering unprecedented capabilities in understanding and generating human-like text. Fine-tuning these models for specific tasks, such as summarization, involves adapting them to the nuances and requirements of the target domain. In this case, the focus was on the summarization of research papers. The research project involved fine-tuning an LLM on a curated dataset of 100 research papers, tailoring the pre-trained model to better align with the specific language, style, and content of academic papers.

To achieve this, Parameter-Efficient Fine-Tuning (PEFT) and Low-Rank Adaptation (LoRA) were utilized. PEFT focuses on modifying a small subset of the model's parameters, making the fine-tuning process more efficient

and less resource-intensive. This approach is particularly beneficial when working with large models and limited computational resources. On the other hand, LoRA involves inserting trainable low-rank matrices into the pre-existing weight matrices of the LLM. By combining these two techniques, the project aimed to optimize the balance between efficiency and performance in the fine-tuning process.

2.1.1 PEFT

Parameter-Efficient Fine-Tuning (PEFT), as introduced by Hugging Face, enables the efficient adaptation of pre-trained language models to specific tasks by adjusting only a limited number of additional parameters. This approach significantly reduces computational and storage costs while maintaining performance levels comparable to those achieved through full fine-tuning.

2.1.2 LoRA

Low-Rank Adaptation (LoRA) enhances fine-tuning efficiency by updating model weights through low-rank decomposition, minimizing changes to the overall parameters and reducing computational load. This innovative approach preserves the original weight matrix, offering benefits such as memory reduction and potentially helping to mitigate catastrophic forgetting. As a result, LoRA has become a popular choice for efficient model fine-tuning.

2.1.3 Fine-tuning

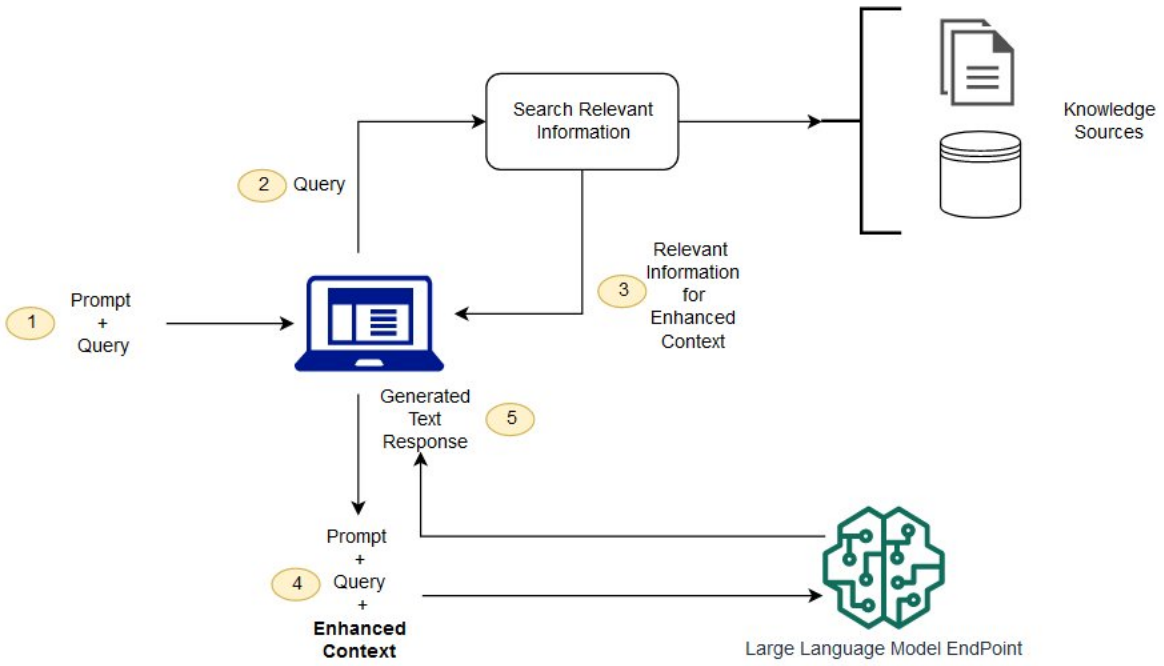
The fine-tuning of the chosen Large Language Model (LLM) was conducted utilizing the NVIDIA A100-SXM4-80GB. The initial phase involved evaluating multiple LLMs to determine the most suitable candidate for the specific task. After thorough testing and consideration, experiments were conducted with Google's FLAN T5, Falcon AI, LLaMA 2 7B, and ultimately, Mistral AI with 7B parameters was selected for its optimal balance of performance and resource usage.
The next step involved configuring the model for fine-tuning, which included the setup of LoRA adapters, essential for the efficient training of the model. To guide the fine-tuning process, the dataset, consisting of research papers and their summaries, was processed. Prompts were engineered to direct the model to summarize the provided research papers. This was achieved by creating a new column in the data frame, formulated as follows:

```
start_prompt = '###Human:\n Summarize the following research paper.\n\n'
end_prompt = '###Assistant:\n\nSummary: '
prompts = start_prompt + text + end_prompt + summary
```

In this formulation, text represents the content of the research paper, and summary is its corresponding summary. These prompts were designed to clearly communicate the task to the LLM, ensuring it understood the objective of summarizing the research papers effectively.
The final stage involved the tokenization of the data and the initiation of the training process. Key training parameters were established as follows: a train batch size of 2, a total of 10 epochs, and a learning rate of 2e-4. Upon the completion of the training, a final loss of 3.98 was achieved. While this value indicates room for improvement, the constraints of time and computational resources necessitated limiting the training to 10 epochs.

2.2 RAG

Retrieval-Augmented Generation (RAG) is a fascinating technique that combines the best of retrieval-based and generative approaches to provide domain-specific applications without requiring fine-tuning. This makes RAG particularly interesting for applications where updating a model frequently with new data or domain-specific knowledge is necessary but computationally expensive.



Retrieval-Augmented Generation (RAG) marks a pivotal innovation in augmenting Large Language Models (LLMs) with external, dynamically updated knowledge bases, diverging from the conventional fine-tuning approach where knowledge is statically integrated into the model. RAG introduces an external "knowledge base" akin to a traditional database, facilitating scalability, reliability, and the integration of up-to-date information. The core of RAG's methodology is the use of source knowledge inputted into the LLM to retrieve relevant data from an external source. This additional information is then incorporated into the model's prompt, enabling the LLM to generate responses that are not only more accurate but also reflect current knowledge.

While "naive RAG" offers a straightforward and efficient implementation, more complex variations exist that can handle nuanced queries with greater precision. Nonetheless, these advanced methods bring trade-offs in terms of response speed and operational costs. The interaction with the LLM required at each reasoning step can be both time-consuming and costly.

A significant benefit of RAG is its ability to reduce hallucinations, a common limitation in modern LLMs where the model generates plausible but incorrect or unfounded information. By grounding responses in retrieved documents, RAG models can offer more accurate and verifiable outputs.

For the vector database essential in the retrieval step, FAISS (Facebook AI Similarity Search) is utilized due to its speed and ease of integration. FAISS can be stored locally, offering an advantage over cloud-based databases by scaling applications to a broader audience without incurring the latency and cost associated with online data retrieval. This local integration of FAISS with RAG not only enhances performance but also aligns with the need for efficient, scalable solutions in deploying advanced NLP applications.

The architecture implemented was a combination of traditional databases with knowledge graphs[7][9] for a Retrieval Augmented Generation (RAG) system. These knowledge graphs are built from the research paper dataset and offer a more organized way to access and use data. This assists the model to pull in rich and related information from these graphs.

For managing these advanced features with our Large Language Model (LLM), we chose the LangChain[6][8] framework. Launched in October 2022, LangChain[6][8] quickly became a go-to tool for developing LLM applications, thanks to its versatility. It's especially useful for tasks like making chatbots, analyzing documents, and summarizing text, which aligned perfectly with our project's needs.

2.3 Traditional Method - Encoder-Decoder with Attention Mechanism

As part of the experiments exploring methods for summarizing research papers, especially in scenarios with limited data availability, another approach investigated was the implementation of a bidirectional LSTM (Long Short-Term Memory) based model augmented with an attention mechanism. This approach leverages the strengths of LSTM units in processing and understanding sequences of text, combined with the ability of attention mechanisms to focus on specific parts of the input text that are most relevant for generating a summary.

Bidirectional LSTMs process the text data in both forward and backward directions, allowing the model to have both past and future context at any point in the sequence. This dual-direction processing significantly enhances the model's ability to understand the context and meaning of the text, which is crucial for tasks like summarization where the relevance of each part of the text can vary greatly.

The addition of an attention mechanism further refines the model's performance by enabling it to selectively concentrate on parts of the text that are most informative for creating the summary. Instead of treating all parts of the input equally, the attention mechanism assigns different weights to different segments of the input, focusing on areas that contribute most to the essence of the content. This targeted approach improves the quality of the generated summaries, making them more coherent and aligned with the main points of the research papers.

This LSTM-based model with attention is particularly beneficial in situations where data is scarce. By efficiently utilizing the available information and focusing on key details, the model can generate concise and meaningful summaries even when the training data is limited. This method represents a valuable tool in the broader toolkit for automating the summarization of academic literature, providing a complementary approach to techniques like Retrieval-Augmented Generation (RAG) and Large Language Models (LLMs).

Model Architecture
The model encoder and decoder was implemented using Long Short-Term Memory (LSTM) network, a type of Recurrent Neural Network (RNN) that is effective in handling sequential data like text. LSTM is an advanced type of RNN that incorporate special units called gates (input, output, and forget gates).These gates regulate the flow of information, allowing the network to retain important past information and forget irrelevant data.This gating mechanism helps LSTM to effectively capture long-range dependencies or context in sequential data, making them effective for complex language tasks like summarizing long texts.

- **Encoder**: The encoder processes the input text and prepares a context-rich representation or context/encoder vector for the decoder.

  Keras LSTM layer was utilized to construct the encoder.The layer was parameterized with settings such as the number of units (which determines the dimensionality of the output space and memory state), activation functions being tanh for the LSTM state (effective in regulating the output values between -1 and 1 aiding in the stabilization of the network's learning) and sigmoid for the gates(effective in controlling the flow of information through the gates by outputting values between 0 and 1).A dropout strategy was also implemented to prevent overfitting by randomly dropping neurons during training.

  The encoder utilizes Keras Bidirectional wrapper, which enables the LSTM layer to process the text in both forward and backward directions. This involves two separate LSTM layers — one that processes the text in the forward direction and another that processes it in the backward direction and the outputs of these two layers are then merged. This approach allows the model to gather a more comprehensive understanding of the text, capturing context from both preceding and following words.

  Keras Input layer was utilized to define the shape of the input data, corresponding to the length of input sequences. This layer specifies the expected size and type of the inputs the model will receive, which is essential to ensure that the LSTM encoder processes fixed-size sequences, aligning with the pre-processed and tokenized format of the input textual data.

- **Decoder**: The decoder generates the summary of the input text.

  Keras LSTM layer was utilized to construct the decoder.In comparison to the encoder, the decoder does not utilize the bidirectional approach as its primary function is to generate text sequentially, where each word prediction uses the previous ones as context.

  The decoder utilizes the context/encoder vector provided by the encoder. This context vector carrying the condensed information of the input text, is used at each step of the sequence generation. Additionally, the

decoder also uses its previous time step outputs as part of its inputs. This combination of the context vector and the previous outputs ensures that each word generated is contextually relevant, both in relation to the original text and the summary generated so far.

Keras Dense layer was utilized for generating the output words.This layer is equipped with a softmax activation function, which converts the LSTM outputs into a probability distribution over the possible output words. The softmax function is ideal as it highlights the most probable word to be chosen as the next word in the sequence, assisting in the generation of coherent and contextually appropriate text.

- **Attention Mechanism**: The attention layer enhances the model's ability to focus on specific parts of the input text while constructing the summary.

  Keras Attention layer is utilized to implement the attention mechanism. This layer is designed to compute attention scores, enabling the model to dynamically focus on different parts of the input sequence.

  The attention mechanism computes a dynamic context vector for each word the decoder generates by assigning weights to different parts of the encoder's output, signifying the relevance of each part of the input text to the current word being generated in the summary. These weights are computed based on the current state of the decoder and the encoder's output, ensuring that the model essentially pays attention to the most relevant parts of the input text, ensuring that it is not only concise but also contextually relevant to the original text.

Training Process

- **Preprocessing and Tokenization**:Utilized Pandas for data loading, text processing and removing non-essential characters such as punctuation, special characters, and numerals, which are generally not useful for text summarization tasks and can introduce noise into the model.Utilized NLTK for tokenization and stop words ('is', 'and', 'the', etc., which do not contribute to the meaning/context of the text) removal.

- **Hyperparameter Tuning and Model Training**: Model compilation involves the selection of an optimizer, loss function, and metrics.For compiling the model, optimizer Adam was chosen as due to its efficiency in handling sparse gradients and adapting the learning rate during training.Categorical cross-entropy was chosen as the loss function as it is a standard choice for classification and in the context of text summarization, it effectively measures the difference between the predicted word distributions and the actual words in the summaries.

  For the model's learning process, three hyperparameters were focused on being learning rate, batch size, and the number of training epochs.The learning rate was adjusted to find the optimal balance between convergence speed and training stability.Adaptive learning rate methods were implemented where the learning rate is initially set higher for faster convergence and then gradually decreased to allow more fine-tuned adjustments as training progresses. Batch size was chosen considering the trade-off between memory usage and model performance. The number of epochs was set to ensure sufficient training without overfitting.For tuning, a baseline number of epochs was set and model performance was monitored on a validation set.Additionally early stopping during training was also implemented.

- **Overfitting Mitigation**: To prevent overfitting, especially due to the limited size of the dataset, regularization technique such as dropout and validation technique such as early stopping were utilized. Dropout was used in the LSTM layers to randomly omit a fraction of the neurons, reducing the model's reliance on any specific set of neurons. In the model, dropout argument was applied to the Keras LSTM layers.The dropout rate was a typical range for dropout rates is between 0.2 and 0.5. Early stopping halts the training process when the model's performance on a validation set stops improving, preventing the model from learning the noise or random fluctuations in the training data. Keras EarlyStopping callback was utilized to implement early stopping mechanism.The callback monitors validation loss, and stops the training process if there is no improvement observed for a defined number of epochs.

3 Experiments/Results

In the course of the experiments, the performance of the fine-tuned model was assessed by comparing the summaries it generated with the original summaries of the research papers. This evaluation process is crucial for understanding how well the model has learned to condense the content of research papers into concise, informative summaries.

3.1 Finetuning Results - Example Summary Comparison

Original Summary: This paper proposes a novel technique to detect anomalies in Electrocardiogram signals and classify cardiac conditions from 45 patients in the MIT-BIH Arrhythmia database. The proposed approach utilizes two methods, one based on conventional Machine learning algorithm (SVM) and the other based on a deep learning method (CNN-based architecture ALEXNET). The deep learning technique improved precision and can be used in clinical settings, with an average classification accuracy of 87.2%.

Predicted Summary (Fine-tuned Model): 1. The paper proposes a comparison of machine learning-based conventional classifier SVM and deep CNN architecture AlexNet for classifying arrhythmia dataset. 2. The raw data is preprocessed, feature extracted, and cleaned data is divided into train and test sets. 3. The SVM model with linear kernel and AlexNet model with optimizer Adam and loss function categorical cross entropy are trained. 4. The future work includes using different deep learning architectures and hyperparameters for ecg signal classification..... (the output still had few more points but for now the above summary does the part)

The provided summaries demonstrate the fine-tuned model's ability to capture key elements of the research paper, although with varying levels of detail and specificity. The original summary succinctly presents the core findings and methodology of the study, focusing on the novel technique for anomaly detection in ECG signals and the comparison between SVM and a deep learning approach using the ALEXNET architecture. It highlights the practical application and effectiveness of the deep learning method, evidenced by the reported average classification accuracy.

The predicted summary by the fine-tuned model provides a more granular overview of the study's process, from data preprocessing to model training, and mentions future work considerations. It reflects the model's capacity to extract and reiterate detailed procedural aspects of the research, though it could be more concise and focused on the study's primary contributions and outcomes.

This comparison illustrates the strengths and areas for improvement in the model's summarization capabilities. The model effectively identifies and communicates essential points, but the inclusion of superfluous details suggests that further refinements could enhance its ability to generate more focused and impactful summaries. This feedback loop is critical for iterative model improvements, aiming for summaries that are not only accurate but also aligned with the conciseness and clarity expected in academic summarizations.

3.2 RAG results

In this section, we demonstrate the capabilities of our Retrieval-Augmented Generation (RAG)[5] system by presenting its responses to specific questions posed about a research paper[12]. These examples highlight the system's understanding and accurate information retrieval capabilities.

Questions and Answers

Question: What is the research on?
Answer: The research focuses on abstractive summarization using a model that incorporates bidirectional RNNs.

Question: Which model did the author propose?
Answer: The author proposed a feature-rich encoder and a switching generator pointer model.

Dealing with Hallucinations

Question: How did Trump contribute to the paper?
Answer: Trump was not mentioned in this paper.

For the question about the research focus, the RAG system accurately identifies the key area of study as abstractive summarization, specifically mentioning the use of bidirectional RNNs. This showcases the system's capacity to pinpoint the main theme of the research without being misled by potentially complex technical details scattered

throughout the paper.

When asked about the model proposed by the author, the system succinctly describes it as a "feature-rich encoder and a switching generator pointer model." This response not only captures the essence of the proposed model but also demonstrates the system's ability to distill detailed information into a concise summary, which is crucial for effectively communicating complex research findings.

The question regarding Trump's contribution to the paper serves as an interesting test for the system's ability to deal with potential hallucinations—a common challenge in language models where the generated content may diverge from factual accuracy. The system's response, "Trump was not mentioned in this paper," is a clear indication of its robustness against incorporating irrelevant or incorrect information, reinforcing its reliability in handling queries with precision.

3.3 Sequence to Sequence model results

In the LSTM-based model, specific patterns were observed in the models performance which highlight both the strengths and limitations of LSTM architectures in handling language tasks like summarization.

The model demonstrated a capacity to identify and extract keywords that are relevant to the domain of the research paper, showing the LSTM's ability to learn from the sequential data, recognizing patterns and important terms within the context of the input text.

The model struggled with arranging these identified keywords into a logically coherent summary, indicating a limitation in LSTM models which is while they are good at capturing local context they can struggle with long-range dependencies and overall text structure.

- **Repetitive and Incoherent Output**: The LSTM model's output demonstrates an issue with token repetition and lack of coherence indicating problems in the model's sequence generation process. While the individual words are relevant to the domain of research text input, they are not arranged in a meaningful way to convey a coherent summary highlighting a deficiency in the model's ability to form logical sentence structures. In LSTM networks, each word prediction depends on the previously generated context. If the model fails to capture the context accurately, it can lead to repetitive or irrelevant word sequences.
- **Training Data Limitations**: LSTM models performance is highly dependent on the training data. If the training data lacks diversity, the model may not learn to generate contextually rich summaries. In comparison, LLMs are trained on extensive data that likely include a variety of specialized texts, enabling them to handle a wide range of topics more effectively.
- **Generalization Challenges**: LSTM models can struggle with generalizing to new types of text if these were not adequately represented in the training data. This is particularly challenging in the domain of research paper summarization, as the language and structure can be highly specialized.In comparison, LLMs benefit from their vast and varied pretraining resulting in superior generalization capabilities.

4 Conclusion

The research project effectively demonstrates the advantages of fine-tuning language models (LLMs), specifically Mistral AI, over traditional encoder-decoder methods in summarizing research papers, especially in scenarios with limited data availability. By employing advanced techniques such as Parameter-Efficient Fine-Tuning (PEFT) and Low-Rank Adaptation (LoRA), the approach successfully generated effective summaries from a dataset of only 100 research papers. Additionally, the efficacy of Retrieval-Augmented Generation (RAG) was showcased in providing insightful and detailed responses without the need for extensive fine-tuning, underscoring its potential in natural language processing tasks, even with minimal data.

This holistic approach combines the strengths of fine-tuned LLMs for generating concise summaries with the dynamic, context-aware capabilities of RAG for answering specific queries. The integration of PEFT and LoRA methods allows for the efficient optimization of the model with minimal computational resources, making it feasible

to enhance the model's performance on specialized tasks like summarization without the need for a large-scale dataset. Meanwhile, RAG's retrieval-based mechanism offers a promising avenue for supplementing the model's knowledge base on-the-fly, facilitating accurate and up-to-date responses that are particularly valuable in fast-evolving fields.

Together, these methodologies pave the way for more scalable, efficient, and adaptable NLP applications, demonstrating the potential for significant advancements in how we process and interact with the vast amounts of information contained in academic literature.

5 References

1. Hu, Edward J., et al. "Lora: Low-rank adaptation of large language models." arXiv preprint arXiv:2106.09685 (2021).
2. Hu, Zhiqiang, et al. "LLM-Adapters: An Adapter Family for Parameter-Efficient Fine-Tuning of Large Language Models." arXiv preprint arXiv:2304.01933 (2023).
3. Wang, Jianguo, et al. "Milvus: A purpose-built vector data management system." Proceedings of the 2021 International Conference on Management of Data. 2021.
4. Jiang, Albert Q., et al. "Mistral 7B." arXiv preprint arXiv:2310.06825 (2023).
5. Lewis, Patrick, et al. "Retrieval-augmented generation for knowledge-intensive nlp tasks." Advances in Neural Information Processing Systems 33 (2020): 9459-9474.
6. Topsakal, Oguzhan, and Tahir Cetin Akinci. "Creating large language model applications utilizing langchain: A primer on developing llm apps fast." Proceedings of the International Conference on Applied Engineering and Natural Sciences, Konya, Turkey. 2023.
7. Yao, Liang, et al. "Exploring large language models for knowledge graph completion." arXiv preprint arXiv:2308.13916 (2023).
8. LangChain Documentation. Available online: `https://python.langchain.com/docs/get_started/introduction` [Accessed on: date].
9. LangChain Blog. "Using a Knowledge Graph to Implement a DevOps RAG Application." Available online: `https://blog.langchain.dev/using-a-knowledge-graph-to-implement-a-devops-rag-application/`.
10. Heidloff, Niklas. "Efficient Fine-Tuning with LoRA." Available online: `https://heidloff.net/article/efficient-fine-tuning-lora/`.
11. Hugging Face. "Mistral 7B Model." Available online: `https://huggingface.co/mistralai/Mistral-7B-v0.1`.
12. Nallapati, Ramesh, et al. "Abstractive text summarization using sequence-to-sequence rnns and beyond." arXiv preprint arXiv:1602.06023 (2016).
13. Poonja, Hasnain Ali, et al. "Evaluation of ECG based Recognition of Cardiac Abnormalities using Machine Learning and Deep Learning." 2021 International Conference on Robotics and Automation in Industry (ICRAI). IEEE, 2021.