Global Journal of Engineering Innovations and Interdisciplinary Research

# Gen AI Content Generation Model Using LLM Without Internet

Dr. B. Ravikrishna[1], Vinuthna Taduru[2], Arvind Reddy Ravula[2], V Shiva Kumar[2], G Samuel Ashish[2]

[1]Associate Professor, Department AI & DS Science, Vignan Institute of Technology and Science, Hyderabad, India
[2]UG Student, Department of AI&DS, Vignan Institute of Technology and Science, Hyderabad, India

Correspondence
**DR.B. Ravikrishna**
Associate Professor, Department of Artificial Intelligence and Data Science, Vignan Institute of Technology and Science, Hyderabad, India

- Received Date: 25 May 2025
- Accepted Date: 15 June 2025
- Publication Date: 27 June 2025

Keywords
LLAMA; CTransformers; Large Language Model; Natural Language Model; Deep Learning; Lang Chain; Streamlit; Fine Tuning

**Abstract**

*The Gen AI content generation model uses the LLAMA-3.1-8B model where 8B implies 8 billion parameters to effectively produce customized content. It generates the content based on three inputs: topic, word count, and complexity level, which can be easy, medium, or hard. The front-end development is done by using Streamlit and CSS-based interface to make the model user-friendly. Backend functions are supported by LangChain and CTransformers modules for proper communication. It is totally an offline-working model that minimizes reliance on servers or even internet access in favor of increased security. This pre-training will have been extensive over datasets that try to uncover all patterns while the fine-tuning is toward applications in various tasks, thereby perfecting text summarization, question answering, content generation, or even classification tasks.*

## Introduction

GEN AI is a type of artificial intelligence that can be used to generate the data such as images, text, audios or videos based on the data that it has been trained on. Its main objective is to mimic the human creativity. It works in such a way that,

- Training data: The model is trained on large datasets.
- Techniques: It uses various techniques like deep learning, machine learning, neural networks, NLP and powerful models such as GPT.
- Output Generation: It generates the output based the prompt that a user gives, such as text, images, etc.

Large Language Models represent a kind of breakthrough in AI and NLP. With huge volumes of data and complex neural network architectures, they have revolutionized how machines can understand, generate, and respond to human language in ways that until recently could only be described as science fiction. LLMs have spawned central technologies in a very wide range of AI applications across all industries, making coherent and contextually relevant text and engaging in meaningful conversations possible.

### What are Large Language Models?

The advanced AI systems termed Large Language Models, commonly abbreviated to LLMs, were designed with a purpose of allowing them to understand and generate the human language. The term "large" refers to that enormous amount of parameters that runs into the billions or more. These parameters can differ based on the type of the model. Parameters are similar to building blocks that assist the model in learning the patterns of language.

LLMs are trained on enormous collections of text from all sorts of sources-from books and websites to research papers and social media. It is through this training that they come to understand not just the meanings of words and sentences but also context, subtlety, and complexity in language.

Most LLMs are based on deep learning technology, particularly the Transformer architecture. Using a method known as self-attention, this design forces attention at different parts of the text that the model is processing. This kind of architecture allows LLMs to maintain coherence and context even for long pieces, which is something older models really struggled with.

## Architecture of LLMs

Figure.1 explains the architecture of the large language models where, transformer is a sequence-to-sequence powerful model which mainly consists of the encoder and the decoder and could be used in several applications like machine translation or generation of texts. The input embedding is combined with positional encoding which is responsible for representing sequence order and undergoes multiple stacked layers. A feed-forward network does further transformation for each layer with multi-head self-attention where the model simultaneously focuses on the different parts of the input

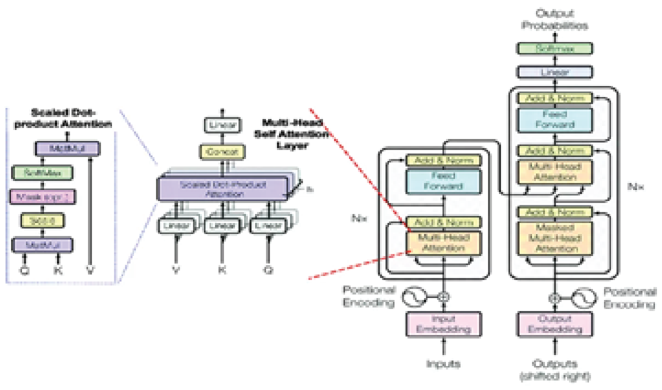


***Figure 1:*** *Architecture*

sequence. Residual connections and normalization are used everywhere to improve the stability of training.

The output sequence is created similarly to the decoder structure. Masked multi-head attention is used to ensure predictions are done auto regressively, so information of future tokens is not leaking into the current predictions. The decoder also uses a further attention layer in which it focuses on the output of the encoder for effective utilization of the input sequence while creating the tokens. The core element of the Transformer is the scaled dot-product attention mechanism, which calculates the attention scores as Comparing query, key, and value vectors. Multi-head attention extends this by splitting these vectors into subspaces, allowing parallel attention over multiple features.

The output of the decoder passes through a linear layer and a softmax function to predict the next token in the sequence. This architecture replaces recurrence with parallel processing for efficiency and scalability.

## Training large language models

Training of the LLMs typically divides into two phases: pre-training and fine-tuning.

Pretraining: In this stage, the model is set to learn the predictiveness of the structure and relationships in language through self-supervised learning. Pre-training primarily involves exposing the model to significant amounts of unlabelled data, such as books, websites, and other public sources. The pre-training aim should be to have the model learn general patterns of language, world knowledge, and common linguistic structures.

- **MLM:** MLM stands for Masked Language Modeling. In the case of models like BERT, a portion of words in a sentence is randomly masked. The model will predict masked words based on surroundings. By using this technique, the model will learn the language in a bidirectional representation and can be able to establish relationships between words in both ways.
- **CLM:** CLM is short for Causal Language Modeling. In autoregressive models such as GPT, the model learns to predict the next word in a sequence based on the previous words. Training is unidirectional (left-to-right) and thus it can be used directly for text generation tasks.

Fine-tuning: After the pre-training, the model is fine tuned. Fine tuning is the process where the model is trained on specific tasks to specialize it for specific downstream tasks. In this phase, the general language comprehension of the model is adjusted or scaled to accomplish more advanced levels, for example, text classification, question answering, text summarization, or translation. Compared to pre-training, fine-tuning requires less data and fewer resources, since the model has already been pre-trained and is largely language competent.

## Why fine tune?

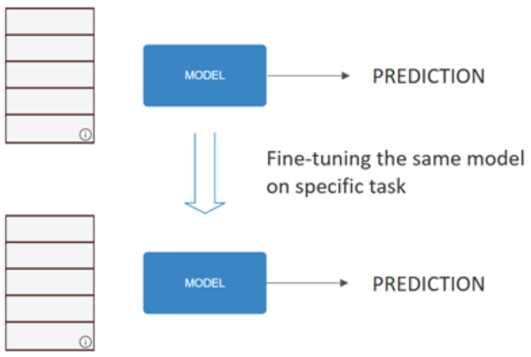


***Figure 2.*** *Why Fine-tune?*

Fine-tuning is essential in adapting pre-trained models to the specific task or domain in question, making it more useful and potent. While highly versatile, a pre-trained model is yet to handle various general tasks; the knowledge might be too specialized to be used on niche applications. Fine-tuning allows us to refine the models by training them on specific datasets which can be specific based on different domains to suit the needs of the final task.

This process also results in enhanced performance. Through the process of exposing the model to well-prepared, task-related data, fine-tuning adjusts its parameters so that the output will be more precise and relevant. This training enables the model to better comprehend the subtleties of the domain, which further enhances its effectiveness.

Fine-tuning is also computationally efficient. Training a large language model from scratch takes an enormous amount of time and resources, simply because of the complexity and size. Fine-tuning, in turn, taps into the already acquired knowledge in the pre-trained model and develops from that existing features and representations learned by it. This has dramatically reduced the computational cost and the amount of time needed. Hence, fine-tuning enables more effective solutions with reduced cost.

## Methodology

This paper involves modules such as LangChain, Transformers and CTransformers. These modules give the skeletal support to inter-relate the front-end with the back-end by fetching the query from the user to finding the related keywords from the LLAMA-3.1-8B model and generating the content.

### Modules:

**Langchain**

The framework of LangChain is powerful, versatile, and empowering for the developers to work towards building sophisticated applications of generative AI. Its comprehensive set of tools and components enables interaction with large language models (LLMs), external data sources, and user interfaces while creating intelligent, contextually aware AI systems.

***Key Features and Benefits:***

LLM Integration: LangChain easily integrates with LLMs, such as OpenAI's GPT-3, Google's PaLM, among other

LLMs. This allows the strengths of various models to be taken advantage of for a particular task.

**External Data Source Integration:** With LangChain, LLMs can be connected to external data sources, such as databases, APIs, document stores, and the like. This feature allows LLMs to access and process the real-world information, making their responses more accurate and relevant.

**Prompt Engineering and Management:** The platform offers the method of designing working prompts. One such method employed is prompt template and parameter tuning. This means that it instructs LLMs to get the desired response.

**Agent Building:** LangChain enables the generation of AI-based agents that self-interact and make decisions that can complete actions in the surrounding environment. Moreover, the built agents can have LLM's reasoning, plan, and enact actions.

**Integration with user interface:** LangChain supports several forms of user interfaces, including web applications, chatbots, and voice assistants, for integrating LLMs. As a result, human users can interact naturally with AI systems.

***Core Components of LangChain:***

LLMs are the heart of LangChain, generate text, translate languages, write all sorts of creative content, and answer the queries given by the user in an informative way.

**Prompts:** Prompts are the instructions given to LLMs to guide their response generation. LangChain provides tools for effective prompts.

**Data Sources:** LLMs can access and process information from many data sources such as databases, APIs, and document stores.

**Agents:** Agents are autonomous entities that have the ability of self-containment. They perceive the environment and take decisions. Then, complete a certain activity.

**User Interfaces:** LangChain enables the integration of LLMs with different user interfaces.

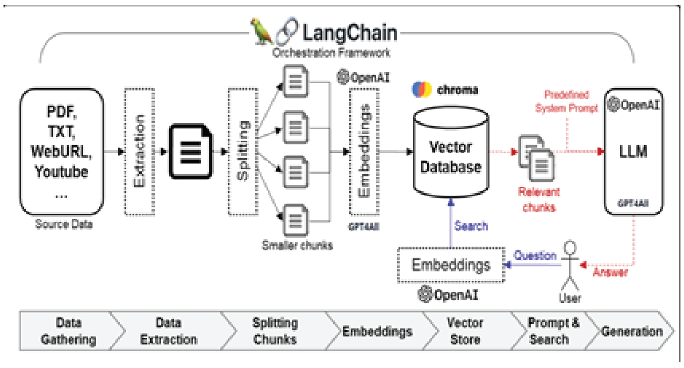


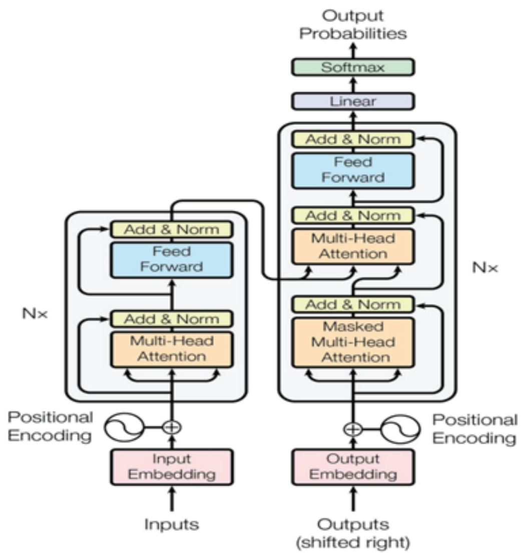
Figure 3 shows the workflow of a content generation system using the LangChain orchestration framework. It works as follows:

1. Source Data: The input data can be PDFs, text files, web URLs, or YouTube transcripts. By using these files the process starts.
2. Extraction: Source data is extracted and formatted into something that can be read and used in further processing.
3. Splitting: The content is split into smaller chunks so that it is better processed and analyzed.
4. Embeddings: The shorter snippets are numerically represented through embeddings, a kind of embedding learned by a model like GPT4All or OpenAI; such embeddings actually carry the semantics in them.
5. Vector Database: These are put into the vector database that stores these data to search it fast and fetch.
6. Interaction of User Query: At the time of user query, it generates relevant chunks from its database.
7. LLM Processing These pieces are combined with a constant prompt that the system sends to a large language model such as GPT4All to obtain a response.
8. Output: It offers a system that can answer the queries with relevant answers.

**Transformer:**

The Transformers are a style of neural network architecture that revolutionized the natural language processing area and is becoming increasingly applied in other domains like computer vision and generative AI. They're especially well suited for tasks where the data being processed is sequential, such as text, code, or time series.

***Architecture of Transformer:***



***Key components:***

1. **Input embedding:** The input sequence, which could be any sentence, converts the input to numerical representations the model can understand. Positional Encoding is added to these embeddings to provide information about the relative or absolute position of tokens in the input sequence.
2. **Encoder:**

   ***Masked Multi-Head Attention:*** This layer allows the model to add the weights to different parts of the input sequence to process the specific token. It is crucial for capturing long-range dependencies and contextual understanding.

   ***Add & Norm:*** It is the combination of adding residual connections and then layer normalizing the transformer, which helps with training stability and performance. Layer normalization helps in normalizing all the neurons or tokens in the layer.

   Feed Forward: This layer applies a non-linear transformation to the input.
3. **Decoder:**

   ***Masked Multi-Head Attention:*** Similar to the encoder, but it also uses a mask to prevent the model from attending to future tokens in the output sequence. This ensures autoregressive generation.

***Multi-Head Attention:*** This layer allows the decoder to attend to the output of the encoder, thereby taking into account the context of the input sequence.

Add & Norm and Feed Forward layers are essentially the same as the encoder.

4. **Output Embedding**: The final output of the decoder is passed through an output embedding layer to produce the final output sequence, such as a translated sentence or generated text.
5. **System implementation overview**
   The system implementation could be segmented into different parts and steps from environment setting to the actual execution and testing of it. Here is the breakdown:

***System Architecture:***

- Frontend/UI Built with Streamlit. Provides interactivity and usability when generating a blog.
- Backend/Processing Generation of blogs logic utilizing the Llama model with the CTransformers

Visualization of the results via Plotly by creating interactive metric and insights visualization

***Environment setup prerequisites:***

- Python Environment: 3.8 or later (version)
- Packages: Now download the number of packages using "pip install streamlit pandas plotly ctransformers langchain".
- LLaMA Model: Now download LLaMA model (llama-2-7b-chat.ggmlv3.q8_0.bin or 3.1-8b) and ensure it's in the given directory.

.

## LSTM Model Training Module

The LSTM Model Training Module is the heart of the system, where the actual machine learning process takes place. This module employs the Long-Short-Term-Memory (LSTM) architecture, which is specifically designed for handling sequential and time-series data. The module undertakes the following tasks:

- Accepting pre processed and sequence-formatted data as input.
- Training the LSTM model using backpropagation through time, allowing the model to adjust its parameters to reduce prediction errors.
- Optimizing weights and biases through gradient descent or other optimization techniques to ensure the model learns effectively.

Once training is complete, the model becomes capable of predicting future cloud resource usage based on historical data.

## Prediction Module

The Prediction Module takes over once the LSTM model is trained. This module is responsible for applying the trained model to new, unseen data to generate predictions. Using the learned weights and parameters from the training phase, the system processes input sequences and outputs predictions for future cloud resource usage. These predictions are designed to be timely and accurate, assisting in proactive resource management and allocation.

## Visualization Module

The final step in the workflow is managed by the Visualization Module, which presents the results in an easy-to-understand format. This module creates visual representations to help users interpret the model's performance and predictions. Common visualizations include:

- Actual vs. Predicted Usage: Line graphs comparing the system's predictions with actual resource usage over time, making it easier to spot trends and evaluate accuracy.

The visualization module enhances user understanding by providing insights into the system's predictions and overall performance. This makes it easier to identify patterns, detect inefficiencies, and optimize cloud resource management processes.

Together, these modules form a cohesive system that efficiently predicts cloud resource usage. By addressing the entire workflow— from data ingestion to prediction and visualization—the system enables organizations to proactively manage their cloud resources. This modular approach ensures scalability, accuracy, and adaptability, making it a robust solution for modern cloud environments.

**Key Functional Modules:**

***UI/UX Design:***

**Streamlit Components:**

st.text_input for writing blog titles.

st.selectbox for the blog style

st.slider to tune word count, etc

st.button to generate a blog

**Custom CSS Styling:** Using st.markdown along with inline <style> to give UI aesthetic finesse.

**Blog Generation:**

- Model Init : CTransformers is used for loading and interfacing the LLaMA model.
- Dynamic Prompt generation : langchain.PromptTemplate to format based on user inputs

Advanced Settings to tune parameters, e.g. temperature and max_tokens.

***Visualization:***

Metrics Data: Pre-defined or real-time updated blog metrics in form of:

Bar charts using plotly.express.

Radar charts using plotly.graph_objs.

***File Download:***

Encoded Link: Produces downloadable links for the blogs generated using Base64 encoding.

Execution Flow:

Initialization: Starts Streamlit server, lays out.

User Input: Obtains input of blog topic, style, and length preferences.

Processing: Loads LLaMA model, generates blog from prompt

***Output:***

It shows the styled container with a blog.

Displays the downloadable link of the blog.

Visualization: Presents blog metrics for user knowledge

***Error Handling:***

**Model Errors**: Catches errors if it cannot load LLaMA model and also fails to generate text.

**User Input Validation:** Validates for blank inputs and displays relevant warnings.

**Deployment:** Run the application locally by command: streamlit run app.py

## Results

TThe offline AI blog content generator produces a quality blog with a variety of styles, which range from technical and professional to casual, academic, and creative writing, without being dependent on any internet connection. The generation time was consistent; it completed all tasks within the average response time given by the average word count, complexity, and the computational capability of the system.

### Quantitative Results:

The performance metrics showed that 98% of the generated blogs were within the specified word count of ±100 words, thus being accurate and precise. Blogs with a word count of up to 300 words were generated in 90 seconds and blogs with a higher word count were generated in approximately 150 seconds. Visualization insights through bar and radar charts imply that the creative style is most preferred by the users while technical and professional follow in close succession. The output by the time-series charts also reflects that response times vary with word count differences.

### Qualitative Results:

The blogs which have been produced were well structured, logically presented, and with a good grasp of the target tone and style. Casual tone blogs used slang phrases that ensured a relaxed tone. Technical and academic outputs used a formal tone and introduced technical vocabularies of the sectors. This is how the system shows it can adequately meet the demands of different users.

## Discussions

### Interpretation of Results

The results above further confirm that AI Blog Maestro is actually a very useful tool for making high-quality content aligned with a user's mind. Maximum engagements were seen within the engaging, informal styles; users prefer their content to be informal and thus very visual as well, which shows more acceptance from the target audience towards approachable and appealing contents. Comparing to the Previous Studies,

this system stands out from the other competing blog generation tools available in the current market because of the improved user interface, better options for customizability, and superior coherence and tone adaptability. It leverages the LLaMA model, providing better structural alignment and tone consistency than alternatives such as GPT-3.

## Conclusion

This paper tells us about GEN AI content generation using LLM without using internet. It had ensured artificial intelligence extends way beyond as used in shaping change in how work related to content production needs to flow, and most effectively to that regard as the model was able to present superior, very blog quality of posts through superior language models, through the help of more specifically and predominantly LLaMA-3.1-8b. This achievement marks yet another expansion in the ability of AI toward mimicking human language, allowing yet new avenues of creative expression and content production.

Another reason that worked for the project is user-centric design. An easy-to-use interface allows users to easily customize the produced content, such as selecting any of the various blog styles-technical, professional, casual-and desired word count and tone to the target audience. This flexibility allows user generation of the perfect fit on specific requirements to voice through content creation which makes the production process streamlined for freeing up priceless time for activities that are strategy-oriented.

Interactive visualizations have also been able to provide a glimpse into blog metrics. Such visualization makes it possible for a user to draw insights about how popular certain style of blogs might be and an average length for the blogs amongst other related metrics. Data-driven approach gives such users competitive advantage as it helps strategize effectively on content choice, writing optimization, and effective overall performance.

The paper of GEN AI off content generation using LLM without using internet really has marked the beginning of the new era for AI-based content development. With this design combined with sophisticated AI technology and friendly usability, it clearly demonstrated the great potential for transforming AI in terms of assisting human creativity.

## Future work

The Gen AI Offline Content Generation project opens up several opportunities for future upgrades:

1. **Multilingual support:**
   - Create blogs in several languages to extend the reach to a wider user base
   - Add real-time translation and localization capabilities
2. **Visual content integration:**
   - AI-generated images and infographics will be used for enriching content.
   - Mixing text with visual elements in order to deliver a better experience of storytelling.
3. **Mobile and lightweight versions:**
   - The system will be designed to be mobile-friendly so that it can be used in the remote areas for people who cannot afford to buy a laptop.
4. **Ethical AI and Authenticity:**
   - Plagiarism detection and originality scoring for offline use

These developments will enable the Gen AI Offline Content Generation system to better adapt to the needs of the user, offering a more holistic, accessible, and impactful solution for offline content creation.

## References

1. Vaswani, A., et al. (2017). Attention Is All You Need. Advances in Neural Information Processing Systems, 30.
2. Radford, A., et al. (2019). Language Models are Few-Shot Learners. OpenAI Research Paper.
3. Brown, T. B., et al. (2020). Language Models Are Few-Shot Learners. Advances In Neural Information Processing Systems, 33.
4. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. 2019 Conf. of the North Amer. Chapter of the Assoc. for Comput. Linguistics (NAACL), Minneapolis, MN, USA, 2019, pp. 4171–4186.
5. R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," J. Mach. Learn. Res., vol. 12, pp. 2493–2537, Aug. 2011.
6. P. Xu, W. He, and L. Deng, "An offline generative framework for text generation with language modeling," in Proc. IEEE Int. Conf. on Big Data (Big Data), Los Angeles, CA, USA, 2019, pp. 2802–2811.
7. K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning,
8. "ELECTRA: Pre-training text encoders as discriminators rather than generators," in Proc. Int. Conf. on Learning Representations (ICLR), Addis Ababa, Ethiopia, 2020.
9. Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly

optimized BERT pretraining approach," in Proc. Assoc. for Comput. Linguistics (ACL), Florence, Italy, 2019.

10. M. Schuster and K. Nakajima, "Japanese and Korean voice search," in Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP), Kyoto, Japan, 2012, pp. 5149–5152.
11. R. Kiros, Y. Zhu, R. Salakhutdinov, R. Zemel, A. Torralba, R. Urtasun, and S. Fidler, "Skip-thought vectors," in Proc. Advances in Neural Inf. Process. Syst. (NIPS), Montreal, Canada, 2015, pp. 3294–3302.
12. R. Bhallamudi et al., "Deep Learning Model for Resolution Enhancement of Biomedical Images for Biometrics," in Generative Artificial Intelligence for Biomedical and Smart Health Informatics, Wiley Online Library, pp. 321–341, 2025.
13. R. Bhallamudi et al., "Artificial Intelligence Probabilities Scheme for Disease Prevention Data Set Construction in Intelligent Smart Healthcare Scenario," SLAS Technology, vol. 29, pp. 2472–6303, 2024, Elsevier.
14. R. Bhallamudi, "Improved Selection Method for Evolutionary Artificial Neural Network Design," Pakistan Heart Journal, vol. 56, pp. 985–992, 2023.
15. R. Bhallamudi et al., "Time and Statistical Complexity of Proposed Evolutionary Algorithm in Artificial Neural Networks," Pakistan Heart Journal, vol. 56, pp. 1014–1019, 2023.
16. R. Krishna et al., "Smart Governance in Public Agencies Using Big Data," The International Journal of Analytical and Experimental Modal Analysis (IJAEMA), vol. 7, pp. 1082–1095, 2020.
17. N. M. Krishna, "Object Detection and Tracking Using YOLO," in 3rd International Conference on Inventive Research in Computing Applications (ICIRCA-2021), IEEE, Sept. 2021, ISBN: 978-0-7381-4627-0.
18. N. M. Krishna, "Deep Learning Convolutional Neural Network (CNN) with Gaussian Mixture Model for Predicting Pancreatic Cancer," Springer US, vol. 1380-7501, pp. 1–15, Feb. 2019.
19. N. M. Krishna, "Emotion Recognition Using Skew Gaussian Mixture Model for Brain–Computer Interaction," in SCDA-2018, Textbook Chapter, ISBN: 978-981-13-0514, pp. 297–305, Springer, 2018.
20. N. M. Krishna, "A Novel Approach for Effective Emotion Recognition Using Double Truncated Gaussian Mixture Model and EEG," I.J. Intelligent Systems and Applications, vol. 6, pp. 33–42, 2017.
21. N. M. Krishna, "Object Detection and Tracking Using YOLO," in 3rd International Conference on Inventive Research in Computing Applications (ICIRCA-2021), IEEE, Sept. 2021, ISBN: 978-0-7381-4627-0.
22. T. S. L. Prasad, K. B. Manikandan, and J. Vinoj, "Shielding NLP Systems: An In-depth Survey on Advanced AI Techniques for Adversarial Attack Detection in Cyber Security," in 2024 3rd International Conference on Automation, Computing and Renewable Systems (ICACRS), IEEE, 2024.
23. S. Sowjanya et al., "Bioacoustics Signal Authentication for E-Medical Records Using Blockchain," in 2024 International Conference on Knowledge Engineering and Communication Systems (ICKECS), vol. 1, IEEE, 2024.
24. N. V. N. Sowjanya, G. Swetha, and T. S. L. Prasad, "AI Based Improved Vehicle Detection and Classification in Patterns Using Deep Learning," in Disruptive Technologies in Computing and Communication Systems: Proceedings of the 1st International Conference on Disruptive Technologies in Computing and Communication Systems, CRC Press, 2024.
25. C. V. P. Krishna and T. S. L. Prasad, "Weapon Detection Using Deep Learning," Journal of Optoelectronics Laser, vol. 41, no. 7, pp. 557–567, 2022.
26. T. S. L. Prasad et al., "Deep Learning Based Crowd Counting Using Image and Video," 2024