



Conversational Image Recognition Chatbot

Devi Praba N¹, Kishor Kumar S², Vijay Surya R³, Ananya R Melagiri⁴,

Vijayaraman S⁵

^{1,2,4,5} 2nd Year Student, Department of Computer Science and Engineering, SSCE

^{3,5} 2nd Year Student, Department of Artificial Intelligence and Machine Learning, SSCE

ABSTRACT

This article proposes how to build conversational image recognition chatbot in python using CNN and LLM. We are going to see about building a model using CNN (Convolutional neural network), LLM (Large language model) and how to implement those with traditional chatbot models like encoders, NLP (natural language processor) and decoders. This chatbot able to detect image and answer the queries, it is can provide answer for variety of question asked on images and objects and it can also answer the simple open ended questions. It is a model where an AI agent can hold meaningful dialogue with humans in natural language about a visual content, specifically images.

1 Introduction

A chat-bot is a software application designed to simulate human conversation and provide automated responses to users' inquiries. Traditional chat bots focus only on text or voice inputs and outputs, limiting their ability to handle visual data. As industries increasingly rely on visuals for operations (e.g., image-based product searches in e-commerce or diagnostic scans in health care), there is a growing need for conversational agents that can interpret and respond to image-based inputs alongside text. Combining chat interfaces with image recognition is particularly useful for enhancing user experience in fields like e-commerce (helping users identify products), health care (aiding in diagnosis with image scans), education (explaining visual data), and more.

Why Build a Conversational Image Recognition Chatbot Using Python? Python is widely favoured in fields such as machine learning, natural language processing (NLP), and computer vision due to its simplicity, extensive libraries, and community support. Building a conversational image recognition chat bot in Python allows developers to integrate powerful image recognition capabilities (using libraries like Tensor Flow, Kera's, or Open CV) with conversational models (like GPT or NLP models in Hugging Face or Open AI). The problem is to design a conversational image recognition chat bot that can understand user-provided images, analyse them using computer vision techniques, and respond contextually, all within a smooth conversation. This requires bridging the gap between two distinct tasks: image recognition and natural language understanding, in a way that is accessible to users through a simple chat interface.



2 Recent works

Visual Dialog Systems (Facebook AI)

- A conversational answer for image input.
- It combines CNNs (image recognition) with language models (NLP).
- Includes the Vis Dial dataset for training models on image-based conversations.

Multimodal AI Models (GPT-4 by Open AI)

- GPT-4 provide support for both text and image inputs allowing a conversational environment on images.
- ChatGPT helps in real-time image understanding and contextual response.

Visual Question Answering (VQA)

- It uses CNN for image recognition when we input the image and answer the queries related to that image.
- VQA v2 Data set is a key resource for developing these systems.

Multimodal Conversational Agents (Google M3)

- It takes inputs from text, images, and video and generate responses that combine all modalities.
- Google Lens integrates image recognition with text-based feedback for visual-based interaction.

Image Captioning Systems (Show, Attend, and Tell)

- It generates descriptive text for images using attention mechanisms in deep learning.
- A crucial step for chat bots to describe images in a conversational manner.

3 Proposed work explanation

CNN

CNN is convolutional neural network. Neural network form the base of deep learning and sub filled machine learning in which algorithms are inspired by human brain. IT is a kind of deep learning model which helps to process images, object detection, and image segmentation. There are many CNN such as LeNet-5, VGGNet, ResNet (Residual Networks) and EfficientNet etc.

EfficientNet CNN

- EfficientNet is a pre-trained CNN model. It is a family of CNN designed to balance model size, accuracy and efficiency. It is used for image classification, especially where efficiency and speed are crucial.
- EffientNet models are known for higher accuracy with fewer parameters and less computational cost compared to other CNN like ResNet or VGG. It uses novel



technique called compound scaling to scale up the model's width, depth, and resolution in balanced way.

- To implement this CNN in python we need to use python libraries such as Tensor Flow/Keras, PyTorch in torch vision library and Timm.

Python code

```
model = models.efficientnet_b0 (pretrained=True)
model.eval()
```

- For pixel-to-pixel tasks like image segmentation architecture designed explicitly for segmentation, such as U-Net, Mask R-CNN or DeepLab commonly used. Efficient can be incorporated into segmentation models by serving as a backbone feature extractor.
- We can combine Mask R-CNN and EfficientNet in chatbot for enhanced performance. We cannot integrate this both CNN in straightforward as both performs different work. We can use two case of Visual Input Understanding and Multimodal interaction. To implement Mask R-CNN in python we need libraries like detectron2 and TorchVision.

LLM

LLM is Large language model which can understand and produced human like response. The large language models are trained on large amount of data and information. It allows them to learn patterns, context and relationships within the languages. We can train our LLM based on our necessity and usage. After initial training we can Fine tune LLM on specific data set to improve their performance. There are many LLM such as GPT-3 and GPT-4(Open AI), BERT (Google), RoBERTa (Facebook AI).

RoBERTa LLM

- RoBERTa (Robustly optimized BERT pretraining approach) is a variant of BERT with some improvements in training strategies, leading to better performance on many language tasks.
- RoBERTa model developed by Facebook AI that build upon BERT, RoBERTa makes several key modifications to BERT's original architecture and training process to improve performance across wide range of NLP(natural language processing) task.
- RoBERTa is also a Masked language model (MLM). The primary purpose of masking is to train the model to predict the hidden parts of contexts.
- It can be implemented in python using libraries like Hugging face transformers and Rasa/ChatterBot. It provides implementation or various version of RoBERTa, such as RoBERTa-base and RoBERTa-large, Pytorch and Tensorflow.

Python code

```
from transformers import RobertaModel, RobertaTokenizer
model = RobertaModel.from_pretrained("roberta-base")
tokenizer = RobertaTokenizer.from_pretrained ("roberta-base")
```



- RoBERTa on its own does not natively generate responses to open-ended questions the way generative model would instead, RoBERTa is primarily used for understanding text and classification tasks where it can be fine-tuned to output a specific category or to retrieve embeddings to represent the context of input text.

Traditional chatbot components

It includes encoders, decoders and NLP which helps to process open ended simple questions. Here encoders and decoders are called sequence-to-sequence (seq2seq) model. These components help the chatbot model to understand user input and give appropriate output. It also contains fall back mechanism, if RoBERTa is unable to answer confidently (e.g., if the confidence score is below a threshold), a fallback mechanism is triggered. The fallback response may include clarifying questions or a response indicating that the bot doesn't understand, encouraging the user to rephrase the question.

Encoders

The encoder gets input from the user as sentence and converts it into a fixed-size numerical representation. This process begins by tokenizing the sentence and converting these tokens into embeddings (numerical vectors) that represent each token in a high-dimensional space. Using neural network encoders processes these embeddings in sequence. The encoder produces the context vector, then it is passed to the decoder.

NLP

NLP is natural language processor. The application of natural language processing techniques is to enable chatbots to understand, interpret, and respond to human language in a conversational way. NLP empowers chatbots to process human language, determine intent, and generate appropriate, contextually relevant responses.

Decoders

The decoder is responsible for generating an appropriate response based on the context vector from the encoder. The decoder that takes the context vector and generates the output sequence. Attention mechanisms may be used to focus on specific parts of the context vector, which improves response quality. The decoder stops generating words when it predicts an end-of-sequence (EOS) token, signifying the completion of the response.

Maintenance of history and memory in chatbot

- EfficientNet CNN and RoBERTa LLM can be integrated into chatbot that remembers the history of questions and answers allowing it to answer follow up questions based on previous interaction.
- To enable memory of previous interactions in our chatbot, we typically need to implant Context management system. Context management system helps chatbot to context storage, context retrieval, dynamic updates and memory management.

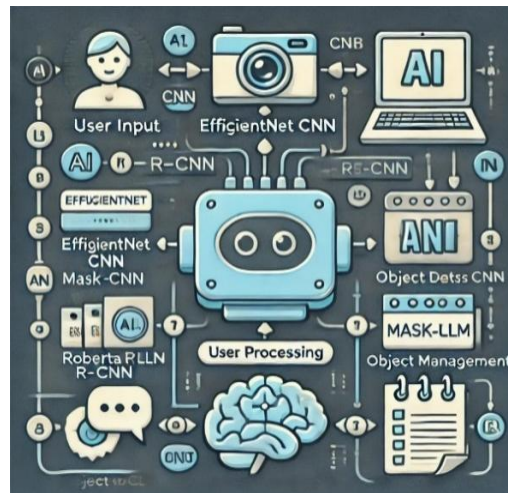


The flowchart of backend process

Backend Process of Conversational Image Chatbot



Here's the flowchart illustrating a chatbot system that combines various AI components, including EfficientNet CNN, Mask R-CNN, and RoBERTa LLM. You can see how each part interacts to process user inputs and manage conversation history.



4 Conclusion

The chatbot which we create in python combining traditional chatbot components with EfficientNet CNN, Mask R-CNN, and RoBERTa LLM offers several advantages for creating a more sophisticated and capable chatbot. Advantages of this chatbot includes, Visual Recognition Capabilities, Improved Accuracy, Enhanced Understanding of Context, Adaptability and Learning, Memory Management, Multimedia interaction. There are some



limitations also which includes Complexity of Implementation, Data Requirements, Interpretability etc. when we work on the limitations and improve the model resource and efficiency, we can create advanced conversational image recognition chatbot.

References

1. Apple (2017). Siri. <https://www.apple.com/ios/siri/>.
2. Google (2017). Google assistant. <https://assistant.google.com/>.
3. Microsoft (2017a). Cortana. <https://www.microsoft.com/en-us/windows/cortana>.
4. <https://web.stanford.edu/>.
5. <https://www.sciencedirect.com/science>