v1: 16 January 2024

Research Article

SnakeChat: A Conversational-AI Based App for Snake Classification

Preprinted: 23 November 2023
Peer-approved: 16 January 2024

Qeios, Vol. 6 (2024)
ISSN: 2632-3834

Jorge Guerra Pires[1]

1. Independent researcher

Conversational Artificial Intelligence (CAI) was the latest big step towards artificial general intelligence (AGI): models of AI that are generic, that can solve problems without being restricted to a small set of problems they were designed for. OpenAI, with ChatGPT, took a big step in this direction. On the other hand, being able to classify snakes from images can help the general public, and even experts, to classify snakes (e.g., finding the right snake serum). There are more than 2,000 different species of snakes worldwide. In this paper, we explore the latest releases of OpenAI APIs. We integrate with a previous work we did, using transfer learning to classify snakes. We have used their GPT Vision API for describing uploaded images textually, their latest ChatGPT API for deciding which model to use for classifying an image, and also to generate a human-friendly final textual response. Additionally, we also tested their GPT-4 API instead of the ChatGPT API as a comparison. We found that the integration was a success. By giving our models to the OpenAI API, it decides which model to call so it could give an educated guess to the user based on an uploaded image of the snake. The best result was when we used the GPT-4 API, something we already expected. We hope that this prototype can be scaled up with more species. We conducted a proof of concept and showed that the basics needed to build such a system for snake classification are possible with current technologies. We hope to contribute to sparking research that brings together computer scientists and biologists, generally called bioinformatics.

Corresponding author: Jorge Guerra Pires, jorgeguerrapires@yahoo.com.br

1. Introduction

When ChatGPT was first released, we could not believe what we were seeing: an artificial intelligence with high-language capabilities; they are called *conversational artificial intelligence*, conversational AI for short. So far, we were used to models that talked in experts' language; we needed those experts to translate their outputs to the public. They were very complex and specialized.

Conversational artificial intelligence (CAI) is a branch of computer science that aims to create systems that can interact with humans using natural language.

CAI systems can be used for various purposes, such as customer service, education, entertainment, health care, and more. CAI systems can be classified into two types, mainly: task-oriented and chat-oriented.

Task-oriented CAI systems are designed to help users accomplish specific goals, such as booking a flight, ordering a pizza, or checking the weather. Chat-oriented CAI systems are designed to engage users in open-ended conversations on various topics, such as sports, movies, or hobbies. Chat-oriented CAI systems are more challenging to develop, as they require a deeper understanding of natural language, human emotions, and common sense.

ChatGPT, as it was first released, is a chat-oriented CAI, the first to gain popularity; in fact, we could manually make it also a task-oriented system since it is a generic model. Recently, OpenAI launched versions of ChatGPT that are also task-oriented CAI; therefore, the line is becoming fuzzy, hard to separate where one type starts and where the other begins. Our system, thanks to this hybrid behaviour of the OpenAI API, is a hybrid CAI (Task-oriented + Chat-oriented). In fact, before ChatGPT, chatbots would require reading books and books, and coding a lot. Now, one can create one, state-of-the-art, in hours or less; see for learning to build one using Wix and ChatGPT API Pires[1].

 doi.org/10.32388/Y13B20.2

We are going to use OpenAI APIs. An API, or application programming interface, is a set of rules and specifications that define how different software components can communicate and interact with each other. APIs are essential for building complex and scalable applications, as they allow developers to reuse existing code and functionality, rather than reinventing the wheel every time. APIs can also enable integration between different systems and platforms, such as web services, databases, cloud computing, mobile devices, etc. APIs can be classified into different types, such as RESTful, SOAP, GraphQL, etc., depending on the format and protocol they use to exchange data; the basic protocol we are using is the JSON format as output from our functions and APIs we use. APIs can also have different levels of security, documentation, and testing, depending on the needs and requirements of the users and developers.

Public API in artificial intelligence was an important paradigm shift. The development and adoption of APIs in artificial intelligence have been an important paradigm shift in the field of computing and software engineering. APIs in artificial intelligence enable developers and users to access and leverage the power of advanced machine learning models and algorithms without having to build them from scratch or understand their inner workings.

OpenAI APIs are quite simple: text in, text out; they have some basic parametrizations but are very simple compared to classical expert-driven APIs. This reduces the cost, time, and complexity of developing and deploying AI solutions and democratizes access to and use of AI across various domains and industries. An API in artificial intelligence also fosters innovation and collaboration, as different AI services can be combined and integrated to create new functionalities and applications. Some examples of popular AI APIs are the Google Cloud Vision API, IBM Watson API, Amazon Rekognition API, Microsoft Cognitive Services API, etc., and more recently, OpenAI APIs. We are actually gathering two artificial intelligence APIs herein: TensorFlow.js from Google, behind SnakeFace, and OpenAI APIs.

We are going to focus on a task-oriented CAI system, as the main goal of our system is classifying snakes by using another system called SnakeFace Pires and Dias Braga[2], which we designed previously using transfer learning. Of course, future versions can be more chat-oriented by providing means for the user to learn more, to ask for more information, to have a "snake-friendly conversation." OpenAI CAI can be a very rich and versatile source for learning more about any topic; their latest model, *gpt-4-1106-preview*, was trained on information until Apr 2023.

Thanks to how the OpenAI APIs are built, it is easy to change the focus of the system. For creating a task-oriented CAI, we are going to explore a new feature of their API called *function calling*: you give the algorithm a set of functions to be used, and the system will decide which one to use. In this approach, as one example, a system could decide to call an external API on traffic to decide whether it should recommend a specific route. In our case, we are going to call external models to confirm a textual description from an image of a snake. This description can be done by the user, or a second model from OpenAI will create the image description using GPT-4, the sequential model of ChatGPT (ChatGPT is powered by GPT-3.5).

Until now, AI models were designed for programmers who knew how to read that 0.8 means with a probability of 80%, and that the model could make mistakes. Artificial intelligence was for experts. ChatGPT made artificial intelligence accessible to the common world. Before ChatGPT, language models were quite limited, such as for content moderation; and large language models were limited to their own companies that developed them. Now, they could create complex and long texts. ChatGPT is what they call a Large Language Model (LLM); their first GPT version included more than 80 GB of texts, which increased in the later versions Kulbik and Saboo[3]. Certainly, people were intrigued by how ChatGPT worked Wolfram[4]. One widely used application is for creating chatbots Pires[1].

Before ChatGPT, Google already had advanced AI APIs made available for programmers, some paid, some free of charge for usage; SnakeFace was built using Teachable Machine, an AI platform using transfer learning made available by Google free of charge. What was done differently by OpenAI is making available an advanced AI model at low cost, a large language model (LLM). OpenAI makes available the pieces one needs to develop their own advanced apps based on AI. They create their own apps but also make available the tools they used. This is a paradigm shift: creating APIs instead of finished software; giving the bricks instead of the pre-built house, build your own house, your own style. This new paradigm creates a new industry: people can create their own advanced AI apps without actually creating an AI model. This is certainly a new way of selling artificial intelligence. It even created what some called "the industry of prompts": people

doi.org/10.32388/Y13B20.2

can make money by creating their own textual commands to ChatGPT and embedding that in a production software Kulbik and Saboo[3].

We suspect, as we used their models, that their biggest revolution was ChatGPT; the sequentials such as GPT-4, which we are going to explore herein, are applications of GPT-3.5, the algorithm behind ChatGPT. For instance, their vision model does not seem to be better than other vision models already available. As we are going to see, it seems to make the same mistakes as other computer vision models we used to do. We suspect they used the same trick we did: feeding a LLM with information from another AI model (e.g., MobileNet). Externally, it may seem like something different, but the trick is that one single big step can serve as leverage to new big steps. Thus, even though some may suggest that GPT-4 sees differently from other computer vision models, we suspect they are using a classical model for vision and connecting it to their language model, which was indeed a revolution. Their revolution was in language models, not in computer vision or other tools they are offering, such as data science (e.g., code interpreter).

One interesting observation we are going to see: they do not seem to be good at "reasoning," handling conflicting information, and arriving at the right conclusion, which can be seen as a *deductive reasoning* capability. As one example, in one response from the OpenAI API, all the correct information was in the response, but the final response was wrong. That was also seen when the image description was not describing a snake species that the model guessed, which would be impossible if one compares the description with the snake species' coloration and patterns. This would be an ability to handle conflicting information and arrive at the right conclusion—something we humans can do easily.

The same behavior was seen in BARD and Bing Chat. It could be explained if we assume those AIs are trained to mine/generate information/text rather than reasoning on information/text. As we are going to see, they do not seem to be good at handling conflicting information from different sources. They may even create a base reasoning for a wrong prediction. Those are certainly undesirable behaviors we need to pay attention to.

A large language model is a type of artificial intelligence system that can generate natural language texts based on a given input, such as a word, a phrase, or a prompt; they can also execute text-related tasks, such as answering questions or summarizing information. Large language models are trained on massive amounts of text data, such as books, articles, websites, and social media posts, to learn the patterns and structures of natural language. Large language models can perform various tasks, such as answering questions, summarizing texts, writing essays, creating captions, and more. However, large language models also have limitations and challenges, such as ethical, social, and environmental implications.

One challenge ChatGPT faced was hallucination[5]: it would make up the answer if it could not answer. One solution that partially solved the problem is allowing ChatGPT to call external sources of information. In some cases, like Bing, it is the browser; for others, it was the powerful software Wolfram[4]. We follow a similar path: we give ChatGPT the ability to call external functions we designed for snake identification. Note that both the SnakeFace and SnakeChat are prototypes; for production usage, they must be scaled up by adding a considerable number of snakes. They are, in their current state, tests of concepts.

In this article, we explore the latest APIs from OpenAI. We have tested how they would support a vision we had in our previous paper, Pires and Dias Braga[2]. We created a set of small models for snake classification and envisioned that ChatGPT would be useful for actually choosing the model to use. Since then, OpenAI launched new models. One of these models, called function calling, is what we needed. Another release we are going to explore is the API version of GPT-4's vision capability. We are going to use it to describe an entered image in case the user does not provide a description for the image. The description is used to decide which model to use (the function calling capability we mentioned).

1.1. Conversational Artificial Intelligence in biology

We conducted research using Semantic Scholar and Bing, powered by ChatGPT, and we were unable to find papers on chatbots in biology. Thus, to the best of our knowledge, we are the first researchers to apply OpenAI APIs to biology in the context of scientific research. However, we will still keep searching and possibly publish a review paper if we find a considerable amount in time. The tendency is that they will become more and more common, as we are doing. Once the cost of implementing those CAIs drops, the entry barrier will also drop.

doi.org/10.32388/Y13B20.2

Building a CAI by itself would require a big team, but thanks to OpenAI, we can just use it, as we are doing herein; it is hard to imagine academic groups having access to such amounts of investment to build such a CAI. That could explain why we were unable to find papers on this topic: it would require a huge team to create the CAI and then apply it to biological cases; the investment barrier was high. This is no longer the case. This is just one book on CAI before OpenAI APIs Freed[6].

Which means we are innovating. We also tried to find CAI in bioinformatics, and the results are focused on medicine. We believe that we may spark research on using those bots in biology, which could help in using those models in biology. Most models can be intimidating, and as we are going to see, we replace all the user interaction with SnakeFace by a chatbot, which we called SnakeChat.

Chatbots can be the UI/UX for the future of biology and medicine. We found in an informal survey that the parametrization of models is a big barrier for medical doctors and biologists when using models Pires[7].

1.2. A brief look at the literature on snake classification

Transfer learning applied to snake classification is an important and promising technique in the field of machine learning Kalinathan et al.[8]; Mirunalini et al.[9]; Yang and Sinnott[10]; Progga et al.[11]; Lakshmi et al.[12]; Krishnan[13]. Several papers have explored this area, highlighting its potential for snake species identification and classification. SnakeFace Pires and Dias Braga[2], a transfer learning-based app, demonstrates the ease of building machine learning models without coding expertise and its adaptability to different biological image identification cases. Another paper proposes a deep learning model that uses transfer learning to build a snake species classifier, combining snake photographic images with geographic location information Desingu et al.[14].

The problem of snake classification is the problem of assigning classes to snakes based on images. Those assignments can be such as their species. This is a problem of *computer vision*. This problem may have several applications, such as educational purposes and finding the proper venom serum. It is crucial to identify the snake correctly, at least with a certain level of accuracy.

Nonetheless, this problem may impose several challenges. We may have images that are not clear or images that are not enough for forming a pattern in the training. In other cases, it is possible that the necessary information is not in the images. See Pires and Dias Braga[2] for more discussions about SnakeFace, one of our base models.

One challenge we are going to face is that as the model becomes more generic with more species on a single model, their precision on species also starts to go down. However, as you have models with high accuracy, they become susceptible to bad usage (e.g., entering an image of a snake not present in the model). And this is a trade-off hard to balance, as we are going to see,

Our model stands out on two main points: it was built in JavaScript and it was built to be used. The models we found in the literature are designed in Python Rajabizadeh and Rezghi[15]; Kalinathan et al.[8]; Yang and Sinnott[10]; Desingu et al.[14]; Durso AMand de Castaneda R[16]; Felipe Guimaraes Dos Santos[17], and they are not available for easy use. This dominance of Python in machine learning is well-known but has been challenged recently by JavaScript. There are several advantages to using JavaScript, the main one being that web applications are built in JavaScript. Thus, one language for the entire process: from machine learning to app development. It may reduce, for instance, costs since you need just one language, and sometimes, even no server (e.g., Node.js or PHP).

If those models in Python were available as APIs, or any other easy form to be used (e.g., NPM packages), we would have been happy to compare our findings with other models for snake classification. For that, we would need to implement them one by one. It does not seem practical. Nonetheless, SnakeFace has been shown to be competitive with those models by Pires and Dias Braga[2]. None of those models, to the best of our knowledge, explore chatbots. Chatbots can make it easier to use models since all they require is a human-like command, and the artificial intelligence will handle the rest.

The reader can find a more detailed discussion about the literature on snake classification in the paper about SnakeFace by Pires and Dias Braga[2].

doi.org/10.32388/Y13B20.2

It is important to mention that iNaturalist, from where we sampled most of our images of snakes, has its own open-source app project, which is a general app. Their project is a community project. I have used their app for suggesting snakes: it tends to be good at associating similar snakes. Nonetheless, it is not good at actually making a final prediction. Initial predictions are later double-checked by humans, their community. The final prediction is a community consensus. Thus, it is an annotation process, more precisely, a crowd annotation. Their platform is perfect for us since the images are real; they are not professional images. The images are the type of images users may take when using our app, amateur images.

1.3. Organization of the manuscript

This article is organized as follows. The next section, section 2, is about the basic tools we have used and explored. Section 3 is about our results and discussions, and how our work fits into the scientific discussions of the OpenAI API new features, our challenges, and what we did well. Finally, in section 4, we present our final words; we close the work by making a short description of what we have discussed and future works, and how our work can be useful. We also provide a list of used references in the reference section.

Two Supplementary Materials are provided: one with the model training details, and the other with a real conversation using the chatbot we designed.

2. Methods

We have used mainly two sets of algorithms for creating SnakeCat: i) SnakeFace by Pires and Dias Braga[2]; and ii) OpenAI APIs. SnakeFace is a transfer learning-based set of models used to classify snakes focused on Brazil; it is a prototype, a concept test. Whereas, OpenAI APIs are a set of APIs focused on large language models (LLMs), with ChatGPT being one of them.

In Section 2.1, the AI that has been used is explored. We made our best effort to pass all the details necessary for reproducibility. In case of interest in reproducing, it would be necessary to learn more about the APIs before trying; we are assuming the reader knows the basics. A tutorial or just the official documentation must be used alongside.

In Section 2.2, the parametrization of SnakeFace is discussed, and details considered important enough were reported.

In Section 2.3, further details are presented that could be essential to understand the paper.

Finally, in Section 2.4, links and external material are provided.

The reader is welcome to get in touch if any information is missing.

2.1. OpenAI APIs explored

See Fig. 4 for a schematic diagram of how those algorithms fit together.

2.1.1. gpt-3-5- turbo-1106

This is the latest version of the ChatGPT API. It will be the default for the ChatGPT API in the future.

2.1.2. gpt-4-106-preview

This is their most advanced model when we wrote this paper. It is their API for GPT-4, released after ChatGPT, with vision capability and more complex reasoning. Several researchers are finding that this version is indeed superior to ChatGPT.

2.1.3. gpt-4-vision-preview

This model is the vision behind GPT 4. When GPT 4 was released, its vision capability was largely commented upon. It can interpret complex images and derive meaning from them. As one example, we have also tested, in collaboration with another author, this algorithm applied to medical images (see section 3.4).

doi.org/10.32388/Y13B20.2

2.1.4. gpt-4-vision-preview

This model is the vision behind GPT 4. When GPT 4 was released, its vision capability was largely commented upon. It can interpret complex images and derive meaning from them. As one example, we have also tested, in collaboration with another author, this algorithm applied to medical images (see section 3.4).

> The image you've provided is a dental X-ray, also known as a radiograph. This type of image is used by dentists to evaluate the health of teeth and the surrounding structures that are not visible during a regular dental examination. In this X-ray, we can see several teeth with their roots. The image shows the internal structure of the teeth, including the pulp chambers and canals. There's also visibility of the surrounding bone. Dentists examine these images for various purposes, such as checking for cavities between teeth, looking at tooth roots, assessing the health of the bony area around the tooth, and checking the status of developing teeth. However, without being a trained dental professional and without more context, it's not appropriate to comment on the specifics regarding the health of these teeth or to diagnose any conditions.

gpt-4-vision-preview reading a medical X-ray.

Sadly, as we have discussed with a medical doctor also doing research on these tools, it cannot make a precise diagnosis. It is similar to snakes: it cannot pinpoint precisely the snake's species. What we have learned is that coupling these models with a more precise model can create something quite advanced. They have released what is called GPTs. It can be trained to better read images; the authors discussed with another author training these GPTs, and the results are promising. Nonetheless, we are going to follow a different step: we are going to use an external library, namely, SnakeFace, which is based on TensorFlow.js.

Section	vision	model selector	final response
section 1	*gpt-4-vision-preview*	*gpt-3.5-turbo-1106*	*gpt-3.5-turbo-1106*
section 2	*gpt-4-vision-preview*	*gpt-4-1106-preview*	*gpt-3.5-turbo-1106*
section 3	*gpt-4-vision-preview*	*gpt-4-1106-preview*	*gpt-4-1106-preview*

Table 1. Different settings for SnakeFace using OpenAI APIs.

2.1.5. Table of results

We have tested combinations of the algorithms we just mentioned in Table 2. The goal is to present how the algorithm will possibly behave with different configurations. This can be easily added as a configuration option from the SnakeChat, with their respective costs.

See Supplementary Material for more examples, using more configurations.

2.2. Parametrization of the algorithms

2.2.1. Parametrization of SnakeFace

We have used SnakeFace basically as reported in Pires and Dias Braga[2]; differences in behavior may exist since we had to retrain the models when writing the paper about SnakeFace, but we have not retrained the models now, when developing SnakeChat; we have just uploaded locally the pretrained models. A pretrained model is a neural network model used after training; it is a model with knowledge. ChatGPT is a pretrained model, as is MobileNet, which is largely used for image identification.

The models used here are below. These links are used to upload the pretrained models locally; this is SnakeFace when we mention it: a set of pretrained models trained to identify snakes. See Pires and Dias Braga[2] for more details. We should stress that these models are prototypes. A production model should have many more snakes to be interesting enough.

1. Fake coral snakes (model 1): https://teachablemac hine.withgoogle.com/models/W9_d1u4f/

2. Fake vs. true coral snakes model (model 2): https://teachablemachine.withgoogle.com/models/9vw2MfLJw/
3. Model with variety (model 3): https://teachablemachine.withgoogle.com/models/Sc8mKQsO/

Tip. If you click on the link, the model will open in the browser, and you can input images.

We are going to use *gpt-3.5-turbo-1106* to automatically decide which models to call; later, we shall also try out *gpt-4-1106-preview*, the GPT-4 API (section 3.3). This is called function calling by the OpenAI team. This function is interesting since you give functions for the chat model to decide whether it should look for external information. For example, if you ask for the temperature in a city, and the model has access to an API about temperature, it can call the API, get the temperature, and give back an educated response; instead of making it up as pure ChatGPT famously does.

This approach of connecting ChatGPT with powerful external tools has been largely explored. For example, the group behind Mathematica, a famous numerical software, used this approach so ChatGPT will give educated responses.

2.2.2. Parametrization of OpenAI APIs

The step-by-step of the algorithm is in Fig. 4. The parametrization at each step is as follows.

1st step: getting a description from the image

Below is the section of the code that calls OpenAI for getting an image description. It is coded in TypeScript and used in Angular. One point: the user can provide a human description as an alternative. The AI-generated description is triggered only when no description is provided by the user. It was also tested using MobileNet, which is a general-purpose computer vision model, for getting a general view of the image (see SM).

```
//It will return a promise, with a call to openAI
return await this.openai.chat.completions.create({
  max_tokens:300,//size of reponse, in tokens
  model: "gpt-4-vision-preview",//model used on the call
  messages: [//Message sent
    {
      role: "user",
      content: [
        { type: "text",
        text:
        ‘There is a snake on this image.
        Describe it.
        Focus on the colors and patterns.
        Ex. the snake is black, red, and white.
        The colors are in strip patterns.
        The black strips appear more easily than
        the red ones.‘ },
        {
          type: "image_url",
          image_url: {
            "url": this.selectedImage,
          },
        },
      ],
    },
  ],
}).then((response)=>{
    //once the promised is finished
  //getting just the response
  return response.choices[0].message.content;
})
```

2nd step: getting which models from SnakeFace to call

The process of calling the OpenAI API is the same as previously. The only change is the parametrization.

```
    const options:any = {
    model: "gpt—3.5—turbo—1106",
    messages: messages,
    tools: tools,
    tool_choice: "auto",
  };
```

gpt-3.5-turbo-1106 is their basic model; it has all the basic ChatGPT capabilities and a large context (the size of possible inputs to the model; basic ChatGPT is about 2 pages).

tools are the SnakeFace functions explained so the model knows what each model does. Explanation given:

1. Fake coral snakes (model 1): Should be used only on snakes that look like coral snakes. This model was trained to identify fake coral snakes: snakes that look like coral snakes. The response is the snake with the highest probability: name and the probability;
2. Fake vs. true coral snakes model (model 2): Should be used only on snakes that look like coral snakes. This model was trained to identify both fake and true coral snakes: snakes that look like coral snakes. The response is the snake with the highest probability: name and the probability;
3. Model with variety (model 3): This model was trained to identify several snakes. It is a generic model for classifying snakes from different species. The response is the snake with the highest probability: name and the probability.

It is important to stress that these texts are called *prompt engineering*. Most of the success in using the OpenAI API is testing these texts. Changes may happen. Generally, we try commands, then we change them looking for desired behaviors. We are constantly changing these texts, looking for better behaviors from the API; it may change in the future.

The links to the models were given previously and used to call them.

3rd step: gathering all the responses and getting a conclusion

This is the trickiest part. Several scenarios may happen. For instance, the SnakeFace prediction may not be in accordance with the openAI API suggestion. This is where those possible conflicting pieces of information may occur. We are constantly testing ways to handle those possible undesirable scenarios.

In this last part, we use *gpt-35-turbo-1106* again for creating a final response. In the final information sent to *gpt-35-turbo-1106*, we have a JSON response from SnakeFace.

```
this.openai.chat.completions.create({
        model: "gpt—3.5—turbo—1106",
        messages: messages,
      }).then((secondResponse)=>{

        this.message_chat.push(
        ‘Em conclus o:
        \n ${secondResponse.choices[0].message.content} ‘);
      }); e
```

Please find the complete method here as a Gist (a GitHub file of code).

2.3. Further details

2.3.1. Using Teachable Machine

SnakeFace was trained using Teachable Machine Pires and Dias Braga[2].

We also wanted to experiment with models that have no overlapping: no overlapping between the species in the models. For the models originally published by Pires and Dias Braga[2], there is overlapping between the models: model 2 contains model 1, and model 3 contains partially model 1 and model 2. In their research scenario, such superposition made sense.

For the case discussed herein, it becomes an issue. The same approach from Pires and Dias Braga[2] was used for our models. Model 1 is the same, except we have retrained the model. We have discarded models 2 and 3 for having overlapping; but their results are still in the paper (Table 2). They were replaced with a model just for true coral snakes, and one model just for the *bothros* genus. The species are the same; we have just reorganized them into separate sub-models, and those sub-models have no species in common. We wanted to investigate the trade-off between large models and specialized models on our chatbot.

The training graphs were added as Supplementary Material.

2.3.2. Model validation

The validation of this system is tricky. Validation is the process by which we make sure our system can achieve its goal on a statistically meaningful level. One possible validation for our case is seeing how well it classifies snakes. The classification is done in three steps: i) get an image textual description (which is provided automatically by the openAI API in case it is not provided by the user); ii) classify by image classification (SnakeFace); iii) use the information so far for making a final guess (openAI API).

The system is composed of two parts: SnakeFace and the openAI API. Therefore, the first step is validating them. SnakeFace was validated in our other paper, Pires and Dias Braga[2], and also in the SM, whereas the openAI API is validated in their respective official documentation; we provide links to the documentation when we use the models. Therefore, the validation of the parts is not our concern; it is out of scope. We assume they were properly done.

The second part that could be validation is how the system works as a whole, the sum of the parts: how the image classification provided by SnakeFace is supported by the openAI API, which is our goal. We believe this validation is meaningless; we shall not do it. The reason is: a single change in either of the APIs we are using will make changes in the SnakeCat behavior, making the validation outdated. OpenAI APIs are constantly updated, and SnakeFace is being developed. Thus, the best we can do is trust their respective validations.

Our validation involves seeing how the system behaves, which we did in the discussion section. We have also provided SM material for several real conversations with comments and comparisons with other alternatives. That is the best validation we believe we can do that makes sense for SnakeChat.

2.3.3. Inputting images

The current model is built to be used by a non-expert user. The model is built on top of other models/APIs. Any image that could be uploaded to a browser will be just fine for classification; you can try to upload the image on our prototype. If the upload option appears to you on the browser, it is fine. All the necessary image preprocessing is either done by us or by third-party APIs.

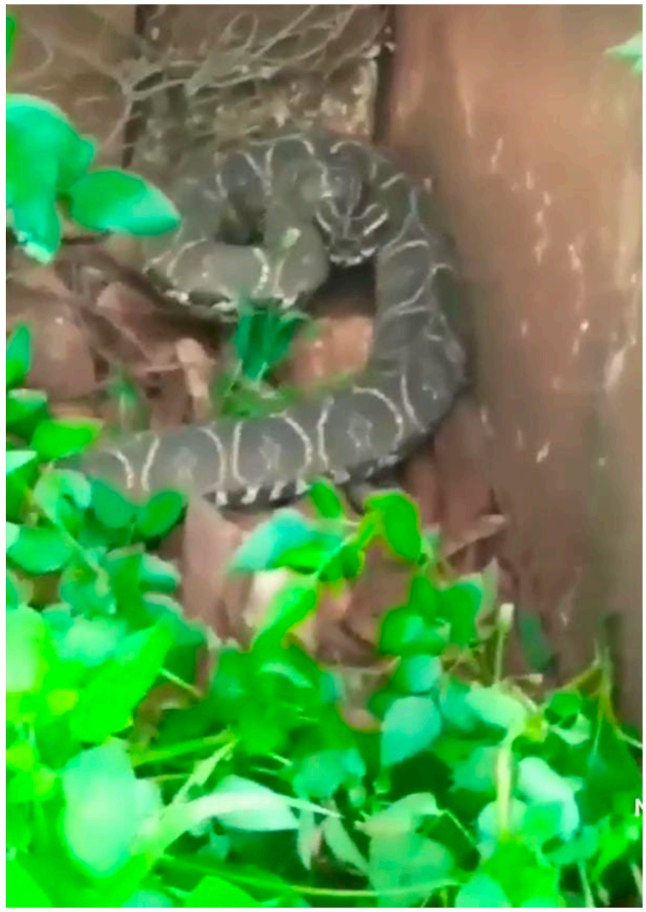

Figure 1. A *bothrops alternatus*, hard to classify. See that during the training, it was possible to classify this image, but as we added new images/species, we lost this good result. The goodness of the model is measured globally, not for a specific image or species. See Pires and Dias Braga[2] for more details. Source: Pires and Dias Braga[2].

The ideal image is a clear image. If you add a non-clear image (e.g., Fig. 1), the model may get confused and misclassify; in some cases, the misclassification can be bad, like classifying *Bothrops alternatus* as *micurus corallinus*. See SM for another example of an image that is hard to classify. Our hope is that GPT λ with its description capability will support this kind of mistake. GPT λ Vision can see and describe Fig. 1 properly. Interestingly enough, SnakeFace, which uses transfer learning, cannot see colors very well. See SnakeFace for more details in Pires and Dias Braga[2], as this is the model that makes the image classification. Regarding the image description by GPT λ, see their documentation for more details.

2.3.4. Transfer learning

Our model is using models that are based on *transfer learning*. How those models are built is discussed in the Method section of Pires and Dias Braga[2]. In this aforementioned paper, it is possible to see the performance metrics for the models. The models that have been used herein were retrained, and most likely, they will be retrained as we add new species and run new experiments. Therefore, it is essential to focus on the qualitative behavior (e.g., whether the accuracy is high enough and the loss function is low alongside the testing curves), not the quantitative (e.g., whether we get a predefined number on accuracy). Once the models are retrained, the focus is on the overall behavior. See Pires and Dias Braga[2] to get familiar with how

doi.org/10.32388/Y13B20.2

those models are trained and what to expect from those training procedures. In the case presented, models already trained were used.

The chatbot discussed herein does not depend on the model used under the hood, nor does it depend on whether a specific technique is used (e.g., transfer learning). Therefore, the decision was made not to focus on how the models under the hood were trained; this is discussed in Pires and Dias Braga[2] and the SM. The reader is gently referred to this paper in case of interest in the models under the hood, which is out of focus for the current paper. Any model can be added under the hood instead of the ones that were used. Any metrics presented herein would most likely be outdated soon since it is planned to add new species and test new approaches, which shall be discussed in future publications. Any new results will not affect the current paper, since the chatbot uses the models, but it does depend on those models for working.

One interesting fact about our approach: "the brain" (i.e., the models) is separated from the interface (i.e., the chatbot).

2.4. External resources: getting to know our prototype

In addition to the links already provided, you can also access a version of the model deployed here. You can upload an image and test it yourself.

Currently, we can classify these species; see SnakeFace for more details in Pires and Dias Braga[2]. When we use the openAI API, we are shifting between those models. The task of the openAI API is creating a conversation and calling the proper model for answering the question; currently, you just upload the image, and the model will decide which model(s) to call and add some possible context. If you upload an image outside of the list, the model will try to fit it on this list: it will take the highest one.

Please, for more details, see Pires and Dias Braga[2]. Also, if you upload a no-snake image, it will still classify. We have tested on SM using MobileNet to avoid these undesirable behaviors. It can be solved by adding a filter on the image workflow, which will prevent no-snake images from entering the classification workflow; MobileNet can do it without any need for training. That is something we intend to work on in the future; it is straightforward to implement (see SM for some initial testing).

2.4.1. model t: just fake coral snakes

Namely: 1) *Apostolepis assimilis*; 2) *Erythrolamprus aesculapi*; 3) *Oxyprhombler*; 4) *Lamporellis triangulum triangulum*.

See SM; this was modified in the current format.

Figure 2. SnakeFace logo.

Lampropeltis triangulum triangulum was eliminated for not being concentrated in our geographical region of interest.

2.4.2. model 2: false coral snakes vs. true ones

In this model, we are using the following true coral snakes: 1) *Micurus corallinus*; 2) *Micurus frontalis*; 3) *Micurus alleni*; 4) *Micurus albicinctus*. For the false ones: 1) *Apostolepis assimilis*; 2) *Erythrolamprus aesculapii*; 3) *Oxyrhopus rhombifier*.

2.4.3. model 3: experimenting with diversity

For the fake coral snakes: 1) *Apostolepis assimilils*; 2) *Erythrolamprus aesculapii*; 3) *Oxyrhopus rhombifer*.

And for the diversity, we have used: 1) *Crotalus durissus*; 2) B*othrops alternatus*; 3) *Bothrops newuied*; 4) *Bothrops jararaca*. Since we already know the difficulties between false and true coral snakes, we left the true coral snakes out.

In the latest model, see SM, we have also added the true coral snake to this model.

3. Results and Discussion

At the end of our last paper, Pires and Dias Braga[2], we proposed an app that would connect SnakeFace with the textual capabilities of ChatGPT (Fig.3).

SnakeFace is a transfer learning-based tool created to classify snakes. Different from general-purpose computer vision models such as MobileNet and the INaturalist Computer Vision, the model was designed to identify Brazilian species. General models, such as MobileNet, tend to overlook Brazilian species; they are good at saying whether we have a snake in the image, even adding a label, but bad at actually finding the species. As we are going to see, the vision model of OpenAI also overlooks Brazilian species. I have also compared it with other models (e.g., BARD from Google, see Supplementary Material). Our goal is to make SnakeFace a part of a conversational AI and still get it right for native species from Brazil. Note that BARD can also work as a general-purpose snake classification chatbot; see SM for a comparison.

If you look closely, our first idealization of SnakeChat (Fig. 3) was by using ChatGPT to guide which algorithm should be used in a possible design with small models (models designed to identify snakes that have something in common, such as being false coral snakes). We had in mind a manual function calling; now OpenAI has made available an option to automatically do that. Furthermore, since we had already done a couple of tests, we knew ChatGPT could get close to guessing snakes by textual descriptions. Then we thought: that could be useful. It is possible that in a real accident with snakes, a textual description is the only information we may have for deciding on the serum.

The motivation for specialized models is that as we grow bigger, the accuracy per class will also decline. When we have general-purpose computer vision models, such as MobileNet, it is impossible to make sure a species is well-classified. In the case of INaturalist, making a mistake will just imply that someone else will correct you. In a real application, such as finding snake serum, a mistake may be costly. Hence, the higher the accuracy per class, the better.

One interesting observation about snakes, compared to other animals, is that they are geographically focused species. If you consider a healthcare center that may want to use the system, it could be served with a local model, trained and adapted for them. Thus, increasing the accuracy and minimizing misclassifications. Even though some species of snakes may have a broad geographical distribution, when we fix a geographical radius of interest, the possible species to be found in that region will drop drastically. Therefore, using general models will just increase the misclassifications, without a real trade-off at stake.

Small models are interesting due to two main reasons: i) when you retrain the model, you do not lose what you did well at the training stage; with a big model, when you retrain, you lose what you did well in the previous training; ii) small models may avoid mistakes between snakes that are very different.

The model does not see as we see; it seems to see globally. The surrounding can deceive the model into classifying badly; one proof of that is that if you remove the background, surprisingly, it does not improve the prediction; it may worsen the prediction. Also, one of the snakes we tested (see Supplementary Material) was identified as a boat by MobileNet, and as a duck by the INaturalist Vision Model. It makes sense since those two misclassifications are generally seen in water, and the snake was in water. Also, a very easy snake to classify was misclassified for being on a human hand (see SM).

One big issue we face with big models is that as we add new species, we lose what we did right. For instance, during the first versions of SnakeFace, we could classify Fig. 1 and Fig. 6. They are two problematic images: one is not clear, and the other is a mimicking snake that does its work pretty well. The model during training is evaluated globally (a global error function is used to guide and finish the training process); it is not evaluated for a specific image. This is necessary since models are trained to operate on uncertainties, not for specific images.

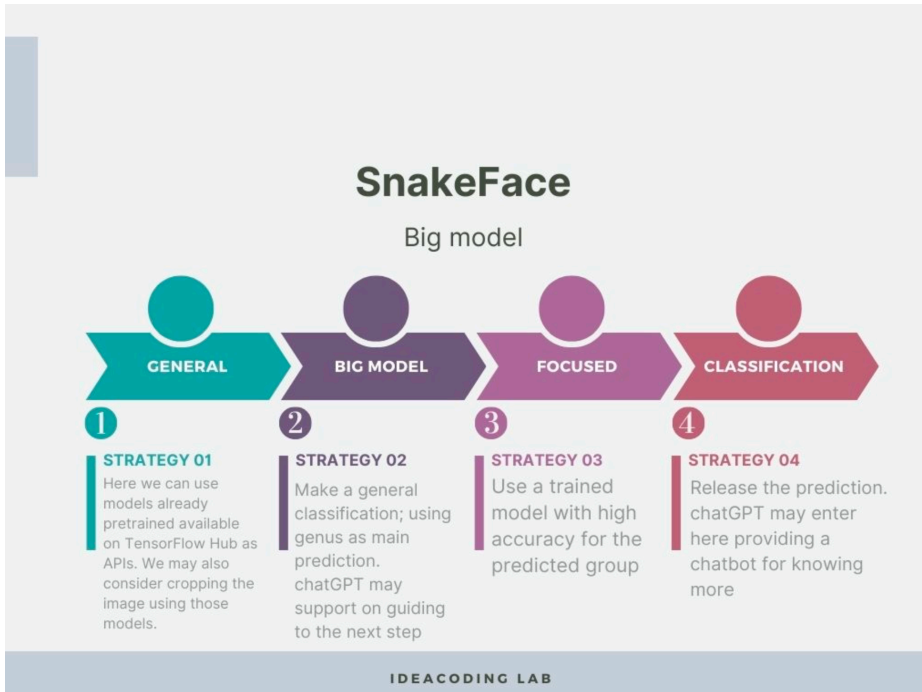

Figure 3. First idealization of snakeChat. Source: Pires and Dias Braga[2].

doi.org/10.32388/Y13B20.2

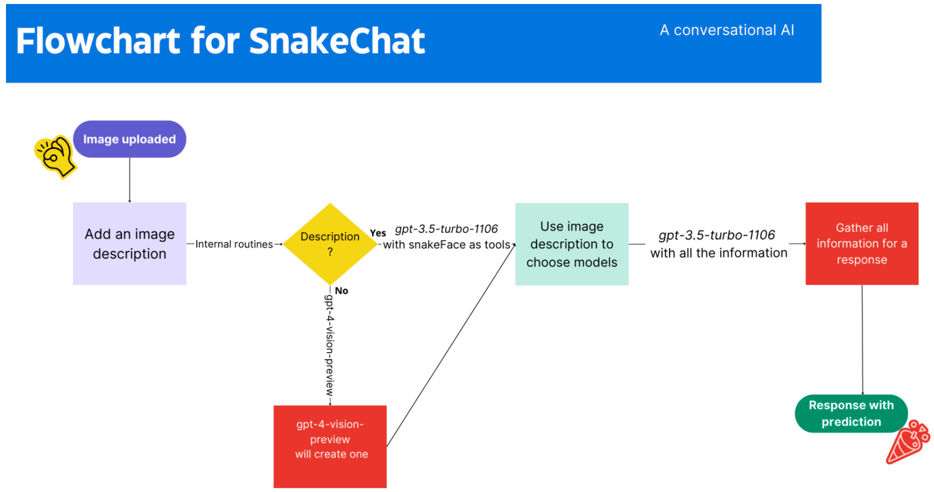

Figure 4. Executed version of SnakeChat based on SnakeFace vision. See that for the last test we present, we have replaced *gpt-3.5-turbo-1106* with *gpt-4-1106-preview*. Legend: the user uploads an image. If the user does not provide an image description, we use the openAI API for creating an image textual description. Once we have the image description, we use the openAI API to decide which model(s) from SnakeFace to use for making a guess using computer vision. Then, finally, we gather all the information (SnakeFace + openAI API) for making a final guess.

Recently, in a single announcement, openAI, the group behind ChatGPT, released new tools for their APIs. Those new releases are what we need to make SnakeChat possible. Before, we were just thinking about ChatGPT as just support and informative for the user. Now, we are actually using their API to make our app smarter. Thus, it was integrated into SnakeFace.

In the following image (Fig. 4), we show how we executed Fig. 3 using as a pivot the latest APIs from openAI. Different from the initial plan to use just the ChatGPT API, we could go further thanks to the latest releases from openAI as APIs.

In the upcoming sections, we are going to discuss more closely three of the results presented in Table 2. The results in this table are for the first version of the chatbot; see SM for more variations of configurations.

See the supplementary material for more examples from the table.

It is important to highlight that this table does not present a statistical analysis. To do so, we would need a considerable amount of work. openAI APIs are constantly updated, and SnakeFace is a prototype; new training will create new emergent behaviors on SnakeChat. SnakeChat is built on top of those models; thus, their behavior will change the behavior of SnakeChat. Thus, at the current moment, we consider any statistical analysis useless, which will also entail financial costs since the openAI APIs are not free of charge. Also, the latest configurations include more options. It is hard to predict how LLMs will behave for each human-text input. We would need to test all the possibilities, variations, and more. We also expect to add more features. The best approach is for each user to play with different settings and see what works best for them.

3.1. Conversations

See SM for more conversations using different configurations.

3.1.1. Oxorphous rhombifer

We are going to start with what the current app does quite well. Rarely, it will misidentify (Table 2).

First, we need to bring to attention, so you can understand better what to expect and from where some undesired behaviors (e.g., stochasticity) may come.

- The algorithms from openAI we are using are stochastic: they offer different responses to the same input. This means that sometimes the response from the openAI API may not be as

doi.org/10.32388/Y13B20.2

helpful as we hope. It seems to improve with their GPT-4 model; see our last conversation;

- SnakeFace is deterministic: the same input will always lead to the same output. This means that it may get it right but be second-guessed by the openAI API. This may occasionally lead to a confusing final response;
- We have not yet figured out a good solution, but you can read the probabilities and pick the highest one. However, for some reason, the openAI API will not necessarily pick the highest probability as the final response. It can be confusing. It seems to trust the textual description more than the numbers. It seems to improve with their GPT-4 model; see our last conversation (see lower sections of Table 2).

Species	SnakeFace	openAI API	Final result
Oxyrhopus rhombifer	got it right when using model for both true and false coral snakes	coral snake (wrong)	Oxyrhopus rhombifer (correct)
Erythrolamprus aesculapii	got it wrong since openAPI API called the wrong model	coral snake (wrong)	*Micrurus frontalis* (wrong)
Oxyrhopus rhombifer	got it right when using both models (just fake corals and both)	coral snake (wrong)	Oxyrhopus rhombifer (correct)
Bothrops Jararaca	got it right, openAPI API called the right model.	caninana (wrong)	*Bothrops Jararaca* (correct)
Erythrolamprus aesculapii	got it wrong since openAPI API called the wrong model	coral snake (wrong)	*Micrurus corallinus* (wrong)
Oxyrhopus rhombifer	got it right with two models, but misclassified on one model	coral snake (wrong)	Micrurus corallinus (wrong)
Bothrops Jararaca	got it right since openAPI API called the right model.	Jararaca (correct)	*Bothrops Jararaca* (correct)

Table 2. Summary of the results we are going to discuss.

***Important**. This is not a statistical analysis; it is a general behavior demonstration.*
***Legend**. The upper section is for gpt-3.5-turbo-1106 as the final model (fourth column) and also as a tool selector. The lower section is for gpt-4-1106-preview as the final model, also as a model selector. The second section (middle) is for gpt-4-1106-preview just at the guessing stage, for selecting which model to use from SnakeFace. See SM for examples of using gpt-4-1106-preview all the way. **Note**. The mistakes of SnakeFace (second column) are influenced when the wrong model is used; that is, they are influenced by the openAI API (third column). The core idea of using the openAI API was precisely to use the right model and avoid the need for a big model. We believe that smaller models will give more control over the possible errors made by SnakeFace. The openAI API can fix the error on a last call (third column), where it should gather all the information and make a guess.*

It is important to bring to attention: variations in the responses are due to openAI APIs.

We have observed two common behaviors: i) when the openAI API helps with the final response; ii) situations where it does not help, creating confusion.

One example is when SnakeFace gets it right, but the openAI API guesses it wrong. Since we are logging all the responses, someone with the ability to handle those contradictions may still gain from the app. Nonetheless, our hope would be an app that handles a single, simple, and correct response. It is not our current case. We saw apparent improvements when using their GPT-4 model: it does not call unnecessary models, therefore, fewer unnecessary numbers to confuse the final response. However, as you can see from Table 2, this behavior from the GPT-4 API is not deterministic; it is not always the same. The lower sections all use the GPT-4 API, so we can have an idea of how it may be used to enhance the SnakeChat behavior.

Another example, see SM, is when a human-entered snake description was provided. The openAI API called the wrong model. By calling the wrong model, SnakeFace provided a wrong prediction, which led to a wrong final response. What is interesting: the GPT-4 API created an entire argumentation to back up the wrong prediction, which was inconsistent with the snake description. It is a sort of "hallucination." I have entered the same conclusion on Bing Chat and Bard; they are both LLMs, and when asked to check for inconsistency, they both failed to see that the snake description was inconsistent with the predicted snake. It is most likely that anyone minimally knowledgeable would see the inconsistency.

One nice feature of the OpenAI API that may come in handy in the future is that they allow training their textual models, called fine-tuning; and then the model will be available to be called. It is possible in the future to try to improve those textual guesses by training their model and using the fine-tuned model instead of the generic one we are using. It is similar to what we did with SnakeFace: we fine-tuned a generic model called MobileNet, which could not identify Brazilian snakes precisely. MobileNet was trained for generic objects. It is safe to assume that in some situations, a text description of the snake is all someone can provide after an accident with the snake, which most likely will run away, or a human-memory description.

Figure 5. Oxyrhopus rhombifer. ***Oxyrhopus rhombifer***, the **Amazon false coral snake**, is a species of snake in the family Colubridae. The species is native to Brazil, Bolivia, Paraguay, Uruguay, and Argentina. Source: Wikipedia.

Our input image is below (Fig. 5): it is a clear image, with no visual distractions, and a simple background. A good question is how the model handles bad images (e.g., Fig. 1): it is uncertain. It may see, or may not see. As SM, there is a case where it is hard even for humans to see the snake hidden in the vegetation, but the model saw it, even though it made a misclassification with a similar snake. Other models, such as MobileNet, failed to see the snake at all. The ideal is always a clear image, with no obstructions.

Keep in mind: it is a matter of training, since the image mentioned was classified in previous training, but we lost it as we added new species. It is something we cannot control.

We are going to test first a human-generated image description. As SM, we have tested in PT since this is the final goal, and we want to make sure the chatbot can handle Portuguese properly.

> The snake is black, red, and white. The colors are in stripes. The black stripes are bigger, whereas the red ones are smaller.

We asked *gpt-3.5-turbo-1106* to guess the snake from the description. It is a parameter for each function; at the current state, it is a dummy parameter, we are just using it as chat logs. It guessed *coral snake*, which is wrong. As said before, the behavior of OpenAI APIs is stochastic; in some tests, they actually guessed it right.

Response when calling the model for both true and false coral snakes.

 doi.org/10.32388/Y13B20.2

> Probilidade de ***82% de ser uma Oxyrhopus rhombifer*** usando SnakeFace, um sistema de identicação por imagens. [correct]

Obs. The system was designed to respond in Portuguese, thus, you will see hybrid responses.

Second call, using the generic model:

> Probilidade de 58% de ser uma **Micrurus corallinus** [this is a true coral snake] usando SnakeFace, um sistema de identicação por imagens.

Remember that the OpenAI API decides which model(s) to call: those calls were done automatically by the OpenAI API. It has at its disposal three models, discussed in Pires and Dias Braga[2]; see the method section for minimal details therein. See SM where we have tested providing more models and more configurations.

If we see the logs, *gpt-3.5-turbo-1106* decided to call two models we made available from SnakeFace: *trueandfalsecorals* and *genericmodelsnake*; the former is used to classify false and true coral snakes, and it was created to try to separate them from each other, see Pires and Dias Braga[2]; whereas the latter is for generic snakes, and it was meant for a possible big model, generic. It did not call for the false coral snake model, which makes sense since its guess was wrong; it was a coral snake. See that in other tests we have done, it actually guessed it was a false coral snake; remember that its behavior is stochastic.

Thus, the first call to SnakeFace rejected *gpt-3.5- turbo-1106* 's guess on the snake, whereas the second guess confirmed it.

However, in this case, it actually joined properly the contradicting information:

> The snake in the photo is most likely an ***Oxyhopus rhombifer***, also known as the Mehuaco or South American False Coral Snake. It is often confused with the ***Micrurus corallinus***, or the Coral snake, which has similar coloration.

This response was perfect: *gpt-3.5-turbo-1106* was able to actually handle the fact that the generic model and *gpt-3.5-turbo-1106* confused the snakes that are alike. The generic model of SnakeFace gave a 15% probability for *Oxyrhopus rhombifer*. The false vs. true coral snake model gave lower than 1% probability of being a *Micrurus corallinus*, which is good. It reinforces our call for specialized small models for SnakeFace instead of big models, as we are proposing herein.

If we make an analogy to a human: *gpt-3.5-turbo-1106* in this case specifically did not fall into confirmation bias. It seems, based on observations during the last

Conversation using *gpt-4-1106-preview*: their GPT 4 model seems to be less likely to fall into confirmation bias, as we use it here (the model uses SnakeFace mistakes to confirm their initial guess).

One may ask ourselves how other researchers are doing. We did a small literature survey in our other paper, Pires and Dias Braga[2]; we are not going to replicate it herein. In essence, there are several researchers also applying transfer learning to snake classification. It seems to be a topic that has gotten researchers interested, and the results are in the same direction as we are going. Except, none of them is using chatbots, or even JavaScript. Those two details may give us an advantage in building an app. Also, they seem to focus on big models. Big models can increase problems with accuracy per class (they may perform well on a set of classes but badly on others). Low accuracy per class can become an issue when it is important to avoid misclassifications (e.g., finding a snake serum).

One issue we found is that those models are built using different models and they are in Python; we are coding in JavaScript. None of those models made available, as far as we know, APIs that we could just plug into our SnakeChat and test. We are open to collaborations with other researchers to test their models inside SnakeChat. All the models used herein were used as legos: we can just remove one and put others for testing (see Fig. 3). This approach makes it easier to compare models, as long as they are easily available for use.

Now, we are going to test the system without a human image description: the AI (*gpt-4-vision-preview*) will make the description.

> The snake in the image has a patterned coloration with three primary colors: black, red, and white. The black color is prominent and appears in thick bands or stripes that run the length of the snake's body. Between the black bands are patches of red

doi.org/10.32388/Y13B20.2

> color, which are themselves outlined with a thinner white or cream border that accentuates the red areas. This tricolor pattern is highly striking and gives the snake a distinct and bold appearance. The arrangement of colors suggests a repeating and systematic pattern, which is often characteristic of certain snake species, particularly those that use their distinct coloration as a warning or as camouflage.

The snake guess was still coral, even with its own image description. We could test fine-tuning their algorithm, something we may test in the future. For this approach, we need image descriptions and their respective snakes. We are not even sure if that would be possible.

In that case, the final response was wrong, even though two of the SnakeFace models predicted the correct snake (*Oxyrhopus rhombifer*): 82% and 69%; one of the models misclassified it as Micrurus corallinus (58%).

But still, the final response was:

> The snake in the photo appears to be a coral snake, specifically a Micrurus corallinus. The tricolor pattern and the prominence of black, red, and white colors are characteristic of coral snakes.

As we said before, someone reading all the logs may say, "it is two against one," and pick the correct guess. Making an analogy to human psychology: here their algorithm fell into confirmation bias.

We tried out their latest model, GPT 4 (*gpt-4-1106-preview*), and also got a similar response:

> Based on the description provided, the snake appears to resemble the coloration and pattern commonly seen in a Coral Snake or a species mimicking a Coral Snake, such as a Milk Snake or a King Snake. [that was a better guess!]
>
> Coral Snakes, which belong to the genus Micrurus or Micruroides in the New World, and the Calliophys, Hemibungarus, or Sionmicrurus genera in the Old World, are known for their distinctive red, black, and yellow/white banding. However, the phrase "red touch black, vnom lack; red touch yellow, kill a fellow" is a mnemonic often used to distinguish the venomous Coral Snake from its non-venomous mimics in some regions (like the Milk Snake or the King Snake) where the order of the colors varies. Nonetheless, this rhyme is not reliable for identifying Coral Snakes globally, as color patterns can differ by region.
>
> Given that the description mentions red and white (or cream), and considering that some Coral Snake species may have a yellowish or whitish band instead of a clear yellow, the snake could potentially be a Coral Snake. On the other hand, since the description does not specify yellow but white or cream, it might be more likely the snake in question

What is interesting about this response is that the algorithm gave the answer when it has written a common saying amongst people: *"red touch black*, *venom lack*; *red touch yellow*, *kill a fellow"*. There is no yellow on the snake's coloration. It finished: *the description does not specify yellow but white or cream, it might be more likely the snake in question*. The conclusion was strange; it seemed it would actually conclude right.

But when we asked to make it short: Coral Snake. Same as the current version we are using. Thus, what remains is trying to fine-tune their model. We are going to leave that experiment for a possible future experiment.

3.2. Erythrolamprus aesculapii

As we brought to attention in our previous work, Pires and Dias Braga[2]: this snake is tough to classify since it was able to fool our models most of the time, mimicking a coral snake. One thing we said in our previous paper: our goal should be avoiding identifying a venomous snake as non-venomous. We called it error type II. In fact, it was the main focus of another research by Felipe Guimaraes Dos Santos[17]. Since we are using SnakeFace, we did not expect different! See that we were able during the training to classify it properly, this image. The problem is that the model is evaluated as good and bad globally, not for a specific image: when you see a good result from the

doi.org/10.32388/Y13B20.2

image, it is an overall model behavior. The version we have used now was retrained after the paper, and it fooled the current model.

We added the description manually for this case:

> The snake is red, white, and black. The colors are in strips. The red strip is the biggest one, whereas the black and white ones seem to be the same size.

We asked also their latest model to guess; it guessed equally wrong compared to the current version we are using from their API. Guessed coral snake (see that this behavior is persistent for coral snakes, see Table 2).

As a result, it decided to use just one model. Guessed as *Micrurus frontalis* (100%); indeed, compared. Fig. 6 and Fig. 7, it is hard to make the difference just using images. We believe that if the user provides details that are not in the image, we believe the whole system (i.e., SnakeChat) can get it right. The issue with that is that just an expert would think to provide information about the head, tail, and more. We have in mind a system for non-experts. We have experimented with asking GPT 4 Vision to focus on these details when generating an image description, and the result is interesting (see SM). When those details are visible in the image, it can describe those parts of the snake, even for images as bad as Fig. 1.

3.3. Bothrops Jararaca

We are going to try something different; we tested as we are, and the guesses were bad, calling *Bothrops jararaca* a coral snake. It may lead to the right answer corrected by SnakeFace and accepted, but also to what we have already seen: a sort of confirmation bias. See lower sections from Table 2.

So far, we used their cheapest model; let‘s try out GPT 4 (*gpt-4-1106-preview*). This is their latest model and widely commented upon, with intriguing results like reading images and graphs with a high level of intelligence.

Figure 6. *Erythrolamprus aesculapii*. Source: Pires and Dias Braga[2].

Figure 7. *Micrurus frontalis*; compare with fig. 6, they are essentially the same visually. Source: Wikipedia.

doi.org/10.32388/Y13B20.2

Figure 8. Bothrops jararaca. Source: Wikipedia.

Description using the same algorithm we have already used so far:

> The snake in the image has a patterned appearance with a mix of neutral and earthy tones. Its scales seem to have a sequence of dark brown or black, beige, and lighter brown hues. The patterns are irregular, with some semblance of bands or blotches that alternate along the length of its body. The darker bands are bolder and more pronounced, providing a type of camouflage among the twigs and leaves on the ground. There do not appear to be any red or white stripes as part of this snake's coloration.

Surprisingly, this description left to guessing as a coral snake with the model used so far. We are changing both the function-calling algorithm and the final response algorithm (see 3; see also Table 2, lower section). Thus, just the image description algorithm was kept; at the moment, it is the only option they provide to describe images.

Their guess was jararaca. Much better. We have done the upload three times, with the same result. We found that the *gpt-4-1106-preview* is more predictable than *gpt- 3.5-turbo-1106*. On their website, they suggest fine-tuning *gpt-3.5-turbo-1106* to get this better result, just using the outputs of *gpt-4-1106-preview* . *gpt-4-1106-preview* is more expensive to use. Regarding the final cost of the fine-tuned model, it does not seem a good deal once the fine-tuned model will be more expensive than using their GPT 4 already trained. It is something we must consider if needed.

Our model confirmed the textual guess *Bothrops jararaca* (97%).

What is interesting is that it also avoided calling unnecessary functions. When guessing as a coral snake, as did the previous setting-up, it would call unnecessary functions, creating noise in the final response: creating space for wrongly guessing the snake. On the calls we did with this image, they always called just one model: the general model; different from the previous setup, which seems to be calling unnecessary functions and creating unnecessary results for the interpreter.

Final response:

> Based on the description provided, the snake in the photo is likely to be a Bothrops jararaca, commonly known as the jararaca. It is a species of pit viper found in South America, and its appearance, characterized by patterned skin with a sequence of dark brown or black, beige, and lighter brown hues, matches the description given. This type of coloration allows the snake to blend into the forest floor, making it well-camouflaged among leaves and twigs. Please note that without the actual photo, this identification should be considered a hypothesis and should be verified with a visual confirmation or by a herpetologist.

doi.org/10.32388/Y13B20.2

Best of all, this result tends to repeat itself if you upload the same image. It is good! See Table 2 for a case where it guessed wrong, but we still got it right.

3.4. How SnakeChat could be adapted to similar applications

Even though we have focused on snakes, this approach can be applied to any case using images. The final result will be the same: a chatbot with state-of-the-art capabilities using images, a *conversational AI*. This includes, as we stressed in our other paper Pires and Dias Braga[2], other species beyond snakes. As long as you can separate the species from the images, you can build a similar system for your desired application.

For instance, we have conducted some tests and discussions with a medical doctor researcher working on those applications as well. You can either train what they call GPTs (which is their way to make their models even more accessible to non-programmers), or use the same path we used here: Teachable Machine, and use the models as we did, as external APIs. We have tested Teachable Machine applied to medical images, and the results are promising. In fact, our next paper may be on a system we are working on, but for medical images instead of snakes. Thanks to how the openAI APIs are built, the transition between applications is straightforward. We have created a first prototype here Pires[18].

Our path has the advantage that Teachable Machine is free of charge to use, all the way, including hosting the model on Google. This means you do not have to pay for the vision part. Also, the GPTs were not available as APIs when we wrote this paper. The computer vision model we are using (namely, *gpt-4-vision-preview*) does not allow fine-tuning, as they pinpoint in their own documentation. We also have, here, a comment on their forum: it may be released in the upcoming future as an API.

Another option is that you have a model from your group, which was well-trained for a specific task. Even though the models from openAI are advanced, they seem to perform poorly on specialized problems. In a couple of tests done by another researcher, and confirmed by us in our own scenarios, if you ask their computer vision models to make diagnoses from images, they will either make a wrong diagnosis or refuse to do so Pires[18]. But they are very good at describing images, as we did. This description can be used to choose more specialized models, as we did.

There is plenty of literature studying the openAI models applied to diagnosis in medicine, their limitations and strengths, as an example, but not exhaustive. Ghosn et al.[19] compared GPT 3.5 and GPT 4.0. As we expected, GPT 4 was better at answering questions from a standard examination in radiology and also explaining the answers.

3.5. Challenges

This section was created based on a public review we received on Qeios.com, on the first draft of this paper. We found the challenges the reviewer raised very insightful and thought it interesting to add those challenges and their respective discussions to this final manuscript. Hopefully, it will support others in understanding our challenges.

3.5.1. Potential for disinformation

There is a risk of disinformation in those LLMs. One example is on the SM, where the LLM created an entire argumentation around a wrong prediction. That is a topic that we may need to handle in the future. Several solutions may be tested. For example, it is possible to provide texts for the models and ask the model to base their answer on this material. That would require a dataset of texts written by experts, which may be a long-term task. Another option is experimenting with fine-tuning their models. These are all possible ways to handle disinformation in those chatbots.

3.5.2. Hallucination

Hallucination is largely associated with LLMs; it can also be seen in computer vision models. For instance, MobileNet saw a boat where we saw a snake, or even a crocodile. See SM. This is something we must consider solutions to mitigate. One way is to avoid using the wrong model. Even though we have tested some possibilities, such as using MobileNet for having a general classification, I am planning to also experiment with other alternatives (e.g., Naturalist).

3.5.3. Integration complexity/dependency on External APIs

Integrating external data sources (APIs), like the Naturalist and OpenAI APIs, as we did and are planning to do, adds complexity to the system. Relying on external APIs, such as OpenAI and INaturalist or any others we may add in the future, introduces a dependency on the availability and stability of these services. Any disruptions or changes in these APIs could impact the functionality of the system. That is indeed a possibility when our system is built using external parts, third-party software/APIs/datasets. It is a trade-off: speed in building comes at a cost, a lack of control over the system's behavior. Moreover, it is hard to imagine that we could build a dataset as rich as Naturalist, where people are constantly posting their observations; they even have a pretrained model on TensorFlow Hub Pires and Dias Braga[2]. In our view, this sharing culture we have built in the last decades is the best thing that ever happened to applied sciences and to collaborative endeavors like ours.

In fact, that was the detail that made a validation senseless, as we did with SnakeFace Pires and Dias Braga[2]: we plotted error graphs, calculated confusion matrices, and more; we did a standard validation of neural network models, following well-accepted metrics and procedures. A single change in any of those external sources (i.e., APIs) will change the validity of the validation.

Systems like ours nowadays fall into the complex system category, a theory largely used in studying biological systems. Complex is not about size when seen from the complex system theory; it is about how hard it is to predict behaviors, as we like to say, "the whole is greater than the sum of its parts," first coined by the philosopher Aristotle. In fact, we have used a term from complex systems theory in this paper; we have used the term 'emergent property' to designate SnakeChat behavior. It is the summation of SnakeFace and OpenAI APIs working synergistically, which by themselves depend on other parts that we cannot see and predict; SnakeFace is more predictable than OpenAI APIs for being a deterministic model.

One big criticism of OpenAI is the fact that their algorithms are closed; no one really knows what is inside the magic hat. They keep a page and an e-mail list where they constantly post instabilities and what they are doing to handle the issues. It is constantly active: it seems they are always making changes to their models, which may create undesirable behaviors. For instance, when we simulated the results in Table 2, we had the impression that the GPT-4 API was more predictable, but now it seems it is behaving as the ChatGPT API, just an impression.

A term that is gaining attention is *software entropy*. This term is used in software testing by Khorikov[20]. We are using Angular as the framework for SnakeChat; Angular was created to be tested, according to Jesse Palmer[21]. Software testing is done precisely to handle, or at least mitigate, software undesirable behaviors, making them more predictable. External APIs and datasets are a big challenge. We do what is called mocks: we just ignore them. We create a local version of APIs and datasets, which behaves in a predictable way, which is unrealistic. The idea is that each group should be responsible for their quality control. Therefore, generally, we do not try to control external APIs/datasets. The best we can do is choose reliable sources. At the current stage, we are not concerned about software testing; we are prototyping. There are whole organizations just focused on software testing.

As we can see from Table 2, the behavior of OpenAI APIs is indeed stochastic, and it seems they are working to increase the predictability of their APIs, which will surely affect ours.

When we conduct a scientific search for terms like "chatbots in biology," most of the results, when we find any, are medicine-driven, and they are using precisely OpenAI solutions. It is impossible to imagine someone building a whole large language model just to classify snakes. The technology behind OpenAI was not created by them; it was around, and it was created by scholars and Google researchers. Therefore, the fact that no scientific group created such an LLM is a sign that it is unfeasible to consider creating our own LLM. The costs of OpenAI are very low, based on what it can do. Thus, it is hard to imagine other groups building such a system with such a low cost. Of course, we do have open-source groups working on that, and their results are interesting. Hopefully, they will also create LLMs, like OpenAI APIs: easy to use, with almost zero setup.

3.5.4. User Understanding and Expectations

This is an interesting point, and we have discussed it previously in Pires[22]. We believe, similar to the mindset from OpenAI, that we learn by interacting with the user. That is why we have

doi.org/10.32388/Y13B20.2

launched all those prototypes. It is hard to handle users' expectations when you are using anything related to ChatGPT, and we have seen that first-hand: all the attention around ChatGPT can backfire; people seem to expect much more than ChatGPT-related systems can deliver. Understanding the user may take time, but we see a general user without expertise in biology. We also see healthcare centers and teaching institutions as final users.

3.5.5. Scalability Challenges

We are not at this stage, but certainly, scalability is something to consider; it is largely studied in start-up contexts, according to Blank and Dorf[23]. One challenge of scalability is adding new species; there are more than 2,000 worldwide, and the best model we know was able to classify about 700 species; we discuss that in our other paper, Pires and Dias Braga[2]. One solution is in this paper: work with small models and use the OpenAI API to handle them intelligently. When we add new species, we cannot know what will happen with the current results. For instance, in early versions of the general model from SnakeFace, we could actually classify both Fig. 1 and Fig. 6, two hard images to classify. As we added new species, we lost this impressive result.

Another challenge, which is discussed in Pires and Dias Braga[2], is annotation, a good dataset. For training, we need images and their respective tags, which have partially been solved using Google Image. Nonetheless, that is something to pay attention to. Our latest models were built using iNaturalist, which is much better. They have a rich community, and annotations are under consensus. Also, their images are real; they are taken by users in real scenarios. This is interesting for training the model.

One detail in our favor: big models tend to lose accuracy per class. We can either build chatbots for regions (e.g., healthcare centers) or change the models according to the user's geographical position. Also, we are considering a model that could better be used to choose small models. Small models could make scalability easier.

3.5.6. Pricing

It is always nice to build such an app for free or at a reduced cost. Nonetheless, there are costs involved that will have to be taken into account in a production model. The OpenAI API has pricing that changes according to the model used. Their pricing is "pay-as-you-go": they charge as you use their models, not a monthly/annual billing. It is hard to make an estimation now since the pricing will change depending on which model is used. For instance, GPT-4 Vision charges higher for high-resolution images. Since we are using more than one model at once, depending on the configuration (see SM), the pricing will vary according to the configuration chosen by the user.

One solution for creating a free version is replacing the function calling by MobileNet, which I have shown not to do a very good job with Brazilian snakes. It will work, but with less accuracy. I have also shown that it can be used instead of the image description model from OpenAI. Again, the result may not be as good as it is now, see SM.

There are several open-source LLMs gaining momentum as the LLMs gain attention for their advanced capabilities, and they may become an option in the future. I need to get more familiar with them and see if it is possible to replace the OpenAI LLMs with them, whenever possible, to reduce costs with LLMs.

There is also the server cost; I am using Heroku. Their pricing is for the entire server; it is possible to deploy several apps on the same plan. For this, options are Wix Site, which allows coding in JavaScript and charges for premium options only. Another option is GitHub Pages. Of course, all those options come with limitations. More options may certainly appear as we go.

3.5.7. Limited Control over Training Data

It is true that we have limited control over what the OpenAI API says. One solution is using a text dataset on snakes we can classify, created by experts. It can be just a blog, since we can access blog posts and send the content to the OpenAI API to use as a reference for their responses. The issue with this strategy is precisely having this expert-curated text dataset. Then, instead of using the knowledge from OpenAI API models, we just use it as a knowledge-mining tool. The latest APIs from OpenAI now can read almost 300 pages at once; they increased considerably their "attention window," a point of large criticism, and also launched Assistants. This is a way to

doi.org/10.32388/Y13B20.2

allow them to actually manage attention windows automatically. This is something we have been exploring largely on other applications where we use chatbots. See a discussion we did on data science Pires[24].

3.5.8. Adding more species

According to *Encyclopaedia Britannica*, there are nearly 3,000 species of snakes distributed nearly worldwide. Brazil is home to a diverse range of snake species, with nearly 400 species found in the country according to the Atlas of Brazilian Snakes. The good question is whether we could scale up. Scaling up is not an issue of SnakeChat, which is a chatbot; it is an issue of SnakeFace Pires and Dias Braga[2], the snake classifier under the hood when the chatbot tries to guess a snake.

Machine intelligence is different from humans': it does not change very much as you add more classes. Humans have a "normal intelligence" (a reference to the normal distribution); it is stronger around expertise, whereas machines have a "flatter intelligence" and tend to be good at a large number of classes at once. It is true that human experts can be better than machine learning in some scenarios; nonetheless, machine learning models, once trained properly, can be good at several cases at once. For classifying several species, one would need several experts. The expert will be good at their expertise but average on the remaining. Machine learning, once trained properly, can be similar to several experts at once.

Even though Pires and Dias Braga[2] explored a small number of species, it was explored strategically: similar snakes and diverse snakes. Those are like extremes. The current paper actually says that we do not need a huge model but a number of small specialized models and to allow the chatbot to pick the proper model.

Thus, adding more species is more an issue of creating more models. Of course, this is time- and resource-consuming, and challenges may appear along the way. See Pires and Dias Braga[2] for more discussions. Scaling this prototype to cover a broader range of snake species will be challenging and will require significantly more data and resources. Nonetheless, this is not an impossible task; we have shown it is possible.

3.5.9. GPT-4 API outperformed ChatGPT API

Recent literature has highlighted the apparent superiority of GPT-4 over its predecessor GPT-3.5, particularly in the context of chat-based language models. Lubiana et al.[25] provide tips for optimizing workflows using ChatGPT, which is powered by GPT-3.5 and GPT-4. Rahaman et al. [26] compare the performance of GPT-4 with previous versions, showing improvements in training data, computation speed, answer accuracy, and overall performance. Rosol et al. [27] reveal that GPT-4 outperforms GPT-3.5 in a medical examination, with higher accuracy. These findings highlight the improved performance and capabilities of GPT-4 compared to GPT-3.5, supporting the claim of its superiority in the literature Lubiana et al.[25]; Rahaman et al.[26]; Rosol et al.[27].

We have mentioned an apparent superiority of GPT-4 over ChatGPT (i.e., GPT-3.5) for our case. A careful statistical analysis has not been done; therefore, it is not safe for us to make any bold claim in that direction. What we can say is that the first simulations showed that GPT-4 tended to call just the needed model, which is good behavior; but in the final simulations, which were done days later, they seemed to be equal in terms of function calling. Calling wrong models, as we have seen, creates noise and unnecessary information, which can induce the final stage of the chatbot to arrive at the wrong conclusion about the snake species. We also cannot say for sure why; nonetheless, we have noticed in another study we have done, which may be published regarding Section 3.4, that the GPT-4 API is more "obedient." Its higher cognitive ability makes it better at deciding which function to call. Thus, the function-calling capability may work better with the GPT-4 API. In fact, it was found that GPT-4 has a higher cognitive capability compared to GPT-3.5 West[28].

3.5.10. Application in healthcare centers

One application I am considering is for the local healthcare center from Antonio Pereira (Ouro Preto). Snakes are local animals. There are groups of studies at the local university (Federal

University of Ouro Preto) to register local snakes; SnakeFace Pires and Dias Braga[2] was created in collaboration with a student from this group. Since accidents with snakes are real, the high precision for a small number of snakes is good. Unless someone is purposely entering a foreign snake, the number of snakes in a geographical location is small. SnakeFace/SnakeChat is good at handling a small number of snakes, different from general models which will misclassify those snakes (see TM). The type of antivenom serum required for treatment depends on the species of snake that caused the bite, and timing may be crucial.

3.5.11. Limitations and future works

In addition to all the testing I have done as part of the development process, I have also provided a set of conversation samples as Supplementary Material. SnakeChat/SnakeFace is not perfect, and it has several flaws that we hope to improve with time.

For instance, as I was finishing the writing of this manuscript version, I found a snake in the same geographical region where the species I added was, during my walks. I knew the species was not in the dataset. After entering the image, the system chose a snake that had a similar color, but a very different species. This happens because, by prompt engineering on the openAI API, I have directly instructed it to use the models even when the snake identified by MobileNet/GPT 4 Vision was different. This was done since initially the function calling would not call when the snake in the image was different from the ones in the models. As an example, MobileNet will identify *Bothrops alternatus* as a rattlesnake, which are very different snakes. The function calling API would refuse to use any model since they are very different snakes.

Thus, we have a tradeoff, which seems without a solution so far: a model was created since the general models we found currently online cannot see the Brazilian snakes well, especially from the geographical region I have concentrated on. However, when a wrong snake is entered, the model would still classify, running the risk of making a totally wrong classification. I have actually tried BARD for the snake mentioned, and it made an even worse prediction. The solution was a manual combination of iNaturalist with Google Image, which I have done manually, and it took me minutes to merge the pieces of information. Therefore, our system can see what it was trained to see where it is, but it can also see what it was trained to see that does not exist (i.e., a snake on wrong models).

This is a general issue with computer vision models, not specific to our case. If you enter an image with just grass and no snake, I have done that as a placebo, see SM, MobileNet saw several animals, including a rabbit. For a bad image, see SM, it even saw a crocodile. In our case, since the system is specialized, it can become easier to see those "hallucinations." I am considering solutions, such as using other general models rather than MobileNet, or GPT Vision. Nonetheless, it is important to keep in mind that some of the issues are bigger than our system; it is a computer vision issue, a general issue. There is no artificial general intelligence currently; all AI nowadays is designed to solve specific problems, as I have done. Some are more specialized (e.g., SnakeFace), others are more generic (e.g., BARD).

4. Conclusion

In this article, we have discussed a prototype for a conversational artificial intelligence (CAI) focused on snake classifications. We have used the APIs from OpenAI as leverage, so we did not have to build a CAI, just use it as leverage. Our model is an extension of SnakeFace, a transfer learning-based model built to identify snakes. SnakeChat is focused on providing a more human-like prediction: instead of just providing a species and probability, we provide it as a textual response. It is a task-oriented CAI, but thanks to how OpenAI APIs are designed, shifting to a hybrid CAI is not that hard, also becoming a chat-oriented CAI.

In addition, we have used textual descriptions that can be entered by the user or generated by an artificial intelligence. Our simulations show a good integration between SnakeFace and OpenAI APIs, which we have named SnakeChat. Our next steps would be: i) adding more species of snakes; ii) adding more information to enhance the predictions; iii) continuing to test our model and keep finding new algorithms that can make our system unique.

One way to add new information to the prediction is by using iNaturalist: they provide an API that can give back information about snakes, such as if they were ever seen at a given location. Several pieces of information, such as textual descriptions, probabilities, and accuracy on a geographical location, can enhance predictions. We believe that GPT-4 can integrate that information into a final response. One path is providing iNaturalist as possible functions to call, something we may test in the future.

We see a platform full of potential, especially after all the advances in artificial intelligence in recent years, both in computer vision and conversational artificial intelligence. The best advances in AI have become public APIs, so they have become bricks for programmers, instead of final tools delivered already designed.

Supplementary Material

This material is available from the Supplementary data section and can be downloaded from the following links:

- snakechat-sm-2.pdf (here)
- snakechat-v4-sm.pdf (here)

Statements and Declarations

Ethics

This study utilized publicly available image data for training snake classification models and publicly available APIs (OpenAI). Image data sources, primarily iNaturalist and Google Images (as detailed in the Data Availability Statement and throughout the manuscript), were accessed in accordance with their terms of service and relevant licenses. The study did not involve direct interaction with human participants or the collection of new data from animals. The medical image discussed in Section 2.1.4 is presented solely for illustrative purposes of the AI's descriptive capabilities on diverse image types and does not represent patient data from this study; appropriate measures for anonymization and consent were not required for its use as an example, as the providers had already anonymized the data before publishing the datasets.

Data Availability

The SnakeFace models analyzed during the current study are available via the links provided in the Methods section (Section 2.2.1). For the underlying image data used to train the SnakeFace models, specific licenses were verified for all images sourced from publicly available repositories (primarily iNaturalist and Google Images) to ensure they permitted reuse for research purposes; these licenses included, for example, various Creative Commons licenses (e.g., CC BY). Further inquiries regarding data can be directed to the corresponding author.

Code Availability

The code developed for the SnakeChat prototype application is available via the deployed prototype referenced in Section 2.4 of the main manuscript. The core method implementation details are available via a Gist link, as provided in Section 2.2.2 of the main manuscript. Further inquiries regarding the code can be directed to the corresponding author.

Author Contributions

Conceptualization, J.G.P.; methodology, J.G.P.; software, J.G.P.; validation, J.G.P.; formal analysis, J.G.P.; investigation, J.G.P.; resources, J.G.P.; data curation, J.G.P.; writing—original draft preparation, J.G.P.; writing—review and editing, J.G.P.; visualization, J.G.P.; supervision, J.G.P.; project administration, J.G.P.

Acknowledgments

This paper was reviewed on Qeios.com, an open-review process. There were more than twenty reviewers. Those reviews were very important for the new version of the paper; they gave me insights on how to improve. Let me thank some of the reviewers publicly, namely: Atheer Alrammahi (University of Al-Qadisiyah); Andi Adriansyah (Universitas Mercu Buana); Muhammad Amir Malik (International Islamic University, Islamabad); Shilpa Nahajan (Dr. B. R. Ambedkar National Institute of Technology Jalandhar); Chirag Joshi; Elakkiya R (Birla Institute of Technology and Science Pilani); Degan Zhang; Venkatesan Venkatesan (Bannari Amman Institute of Technology); Rodrigo Marques de Figueiredo (Universidade Vale do Rio dos Sinos);

Muna Hadi Saleh (University of Baghdad); Mei Hua Hsu (Chang Gung University of Science and Technology); Newlin Shebiah Russel *et al.*

References

1. a, b *Pires JG (2023b). From the donkey to the chatty Einstein: using ChatGPT to create an intelligent virtual assistant using the Wix platform. Amazon.com.br. https://www.amazon.com.br/Jumento-Einstein-Chatty-assistente-inteligente-ebook/dp/B0C1P811QF.*
2. a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, ^, ^, ^, ^, ^, ^, ^, ^, ^, ^ *Pires JG, Dias Braga LH (2023). "Snakeface: a transfer learning based app for snake classification." Revista Brasileira de Computação Aplicada. **15**(3):80–95. https://seer.upf.br/index.php/rbca/article/view/15028.*
3. a, b *Kublik S, Saboo S (2022). GPT-3: Building Innovative NLP Products Using Large Language Models. illustrated ed. O'Reilly Media, Incorporated.*
4. a, b *Wolfram S (2023). "What is ChatGPT doing . . . and why does it work?." writings.stephenwolfram.com. https://writings.stephenwolfram.com/2023/02/what-is-chatgpt-doing-and-why-does-it-work/.*
5. ^ *Emsley R (2023). "ChatGPT: these are not hallucinations – they're fabrications and falsifications." Schizophrenia. **9**(52):1–2.*
6. ^ *Freed A (2021). Conversational AI: Chatbots that work. Manning. https://www.amazon.com/Conversational-AI- Chatbots-that-work/dp/1617298832.*
7. ^ *Pires JG (2021). "An informal survey presents the gap between computer and medical doctors and biologists." Medium. https://medium.com/theoretical-and-mathematical-biology/an-informal-survey-presents-the-gap-between-computer-and-medical-doctors-and-biologists-ca8816051739.*
8. a, b *Kalinathan L, Balasundaram1 P, Ganesh P, Bathala SS, Mukesh RK (2021). "Automatic snake classification using deep learning algorithm." Conference and Labs of the Evaluation Forum. https://ceur-ws.org/Vol-2936/paper-135.pdf.*
9. ^ *Mirunalini P, Desingu K, Bharathi H, Chodisetty EA, Bhaskar A (2022). "Deep learning and gradient boosting ensembles for classification of snake species." Conference and Labs of the Evaluation Forum. S2CID 251470924.*
10. a, b *Yang Z, Sinnott RO (2021). "Snake detection and classification using deep learning." Proceedings of the 54th Hawaii International Conference on System Sciences. 1212–1222. https://scholarspace.manoa.hawaii.edu/server/api/core/bitstreams/f9f2de4d-ddaa-4984-9fc3-9bbca4e27247/content.*
11. ^ *Progga NI, Rezoana N, Hossain MS, Islam R ul, Andersson K (2021). "A cnn based model for venomous and non-venomous snake classification." Analogical and Inductive Inference. S2CID 236963195.*
12. ^ *Lakshmi D, Panda RC, Amrita, Prakash A (2021). "Life-saving app: Snake classification 'venomous and non-venomous' using fast.ai based on indian species." Artificial Intelligence Systems and the Internet of Things in the Digital Era. S2CID 236671950.*
13. ^ *Krishnan MG (2020). "Impact of pretrained networks for snake species classification." Conference and Labs of the Evaluation Forum. S2CID 225073809.*
14. a, b *Desingu K, Mirunalini P, Kumar J (2021). "Snake species classification using transfer learning technique." Conference and Labs of the Evaluation Forum. S2CID 237298537.*
15. ^ *Rajabizadeh M, Rezghi M (2021). "A comparative study on image-based snake identification using machine learning." Scientific Reports. **11**(1). doi:10.1038/s41598-021-96031-1.*
16. ^ *Durso AM, Moorthy GK, Mohanty SP, Bolon I, Salathé M, Ruiz de Castañeda R (2021). "Supervised Learning Computer Vision Benchmark for Snake Species Identification From Photographs: Implications for Herpetology and Global Health." Frontiers in Artificial Intelligence. **4**. doi:10.3389/frai.2021.582110.*
17. a, b *Dos Santos FG, Machiaveli EA, M JMV (2021). "Snake classifier: Mobile application for the classification of venomous snakes." Academia.edu. https://www.academia.edu/95702218/Snake Classifier _Aplicativo mobile para classifica%C3%A7%C3%A3o de serpentes pe%C3%A7onhentas.*
18. a, b *Pires JG (2023c). "Robodoc: a conversational-ai based app for medical conversations." medRxiv. 2023:2023.12-31.23300681. https://www.medrxiv.org/content/10.1101/2023.12.31.23300681v1.*
19. ^ *Ghosn Y, Sardouk OE, Jabbour Y, Jrad M, Kamareddine MH, Abbas N, Saade C, Ghanem AA (2023). "ChatGPT 4 versus ChatGPT 3.5 on the final frcr part a sample questions. assessing performance and accuracy of explanations." medRxiv. https://www.medrxiv.org/content/early/ 2023/09/08/2023.09.06.23295144.*

20. [^]Khorikov V (2020). Unit Testing Principles, Practices, and Patterns. Manning Publications.
21. [^]Palmer J, Cohn C, N MGC (2018). Testing Angular Applications. Manning Publications. https://www.manning.com/books/testing-angular-applications.
22. [^]Pires JG (2022). "Innovating with Biomathematics: the challenge of building user-friendly interfaces for computational biology." Academia Letters. Article 5792. doi:10.20935/AL5792.
23. [^]Blank S, Dorf B (2012). The Startup Owner's Manual: The Step-By-Step Guide for Building a Great Company. K & S Ranch.
24. [^]Pires JG (2023a). "Data science using openai: testing their new capabilities focused on data science." Qeios. https://www.qeios.com/read/76QMHB.
25. [a, b]Lubiana T, Lopes R, Medeiros P, Silva JC, Gonçalves ANA, Maracaja-Coutinho V, Nakaya HTI (2023). "Ten quick tips for harnessing the power of chatgpt/gpt-4 in computational biology." Semantic Scholar. S2CID 257805086.
26. [a, b]Rahaman MS, Ahsan MMT, Anjum N, Terano HJR, Rahman MM (2023). "From ChatGPT-3 to gpt-4: A significant advancement in ai-driven nlp tools." Journal of Engineering and Emerging Technologies. S2CID 258650483.
27. [a, b]Rosoł M, Gąsior JS, Łaba J, Korzeniewski K, Młyńczak M (2023). "Evaluation of the performance of gpt-3.5 and gpt-4 on the medical final examination." medRxiv. S2CID 259073477.
28. [^]West CG (2023). "Advances in apparent conceptual physics reasoning in gpt-4." ArXiv. abs/2303.17012. S2CID 257913307.

Supplementary data: available at https://doi.org/10.32388/Y13B20.2

Declarations

Funding: No specific funding was received for this work.

Potential competing interests: No potential competing interests to declare.