

Research Article

SnakeChat: a conversational-AI based app for snake classification

Jorge Guerra Pires¹¹ Institute for Development and Economic Analysis, Yogyakarta, Indonesia

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 gave a big step towards 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. On 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 openAI API, it decides which model to call as so it could give an educated guess to the user based on an uploaded image from 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 did a test of concept, and showed that the basics needed to build such a system for snake classification is 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

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 to the public their outputs. 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 firstly released, is a chat-oriented CAI, the first to gain popularity; in fact, we could manually make it also a task-oriented 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 starts one type and where begins the other. Our system, thanks to this hybrid-behaviour of openAI API, is a hybrid CAI (Task-oriented + Chat-oriented). In fact, before chatGPT, chabots would require readings 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 (2023).

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 protocols we are using is 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 API in artificial intelligence has been an important paradigm shift in the field of computing and software engineering. API in artificial intelligence enables 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 the access and use of AI across various domains and industries. 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 Google Cloud Vision API, IBM Watson API, Amazon Rekognition API, Microsoft Cognitive Services API, etc. and more recently, openAI APIs.

We are going to focus on a task-oriented CAI system, once the main goal of our system is classifying snakes, by using another system called SnakeFace Pires and Braga (2023), that 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. On 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 from 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 to gain the common world. Before chatGPT, language models were quite limited, such as 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 (LLMs), their first GPT version included more than 80 GB of texts, which increased in the later versions Kublik and Saboo (2022). Certainly, people were intrigued by with how chatGPT worked Wolfram (2023). One widely applications is for creating chatbots Pires (2023).

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 different by openAI is making available an advanced AI model with 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 making available the tools they used. This is a paradigm shift: creating APIs instead of the finished software; giving the bricks, instead of the pre-built house, build your own house, you 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 can make money by creating their own textual commands to chatGPT, and embed that on a production software.

We suspect, as we used their models, that their biggest revolution was chatGPT; the sequentials such as GPT 4, that 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 do. We suspect they used the same trick we did: feed a LLM with information from another AI model (e.g., MobileNet). Externally, may seem like something different, but the trick is one single big step can serve as leverage to new big steps. Thus, even though some may suggest that GPT 4 sees different from other computer vision models, we suspect they are using classical model for vision, and connecting to their language model, which was indeed a revolution. Their revolution was on language models, not on computer vision, or other tools they are offering, such as data science (e.g., [code interpreter](#)).

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 Emsley (2023): 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, as Bing, it is the browser, for others, it was powerful software Wolfram (2023). We follow a similar path: we give chatGPT the ability to call external functions we designed for snake identification. See that both the

SnakeFace and SnakeChat are prototypes, for production usage, they must be scaled up, adding a considerable amount of snakes. They are at the current state tests of concepts.

On this article, we explore the latest APIs from openAI. We have tested how it would support on a vision we had on our previous paper Pires and Braga (2023). We have created a set of small models for snake classification, and we envisioned that chatGPT would be useful for actually choosing the model to use. Since then, openAI launched new models. One of this models, called function calling, is what we needed. Another release we are going to explore is the API version of GPT 4 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).

This article is organized as following. The next section, section 2 is about our basic tools we have used, and explored. Section 3 is about our results and discussions, and how our work fit in on the scientific discussions of the openAI API new features, our challenges and what we did well. Finally, on 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 on the reference section.

2. Methods

We have used mainly two set of algorithms for creating SnakeChat: i) SnakeFace Pires and Braga (2023); 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 testing. Whereas, openAI APIs is a set of APIs focused on large language models (LLMs), being chatGPT one of them.

2.1. openAI APIs explored

See fig. 3 for a schematic diagram on 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 chatGPT API in the future.

2.1.2. gpt-4-1106-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, e.g. Ghosn et al. (2023).

2.1.3. gpt-4-vision-preview

This model is the vision behind GPT 4. When GPT 4 was released, their vision capability was largely commented upon. It can interpreter complex images and get 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 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 those tools, it cannot make a precise diagnosis. It is similar to snakes: it cannot pinpoint precisely the snake's species. What we have learnt is that coupling those 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 those 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 in TensorFlow.js.

2.2. Parametrization of the algorithms

2.2.1. Parametrization of SnakeFace

We have used SnakeFace basically as reported on Pires and Braga (2023); differences in behaviors 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 it is MobileNet largely used for image identification.

The models used here are below. Those 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 Braga (2023) for more details. We should stress that those models are prototypes. A production model should have much more snakes to be interesting enough.

- i. Fake coral snakes (model 1): https://teachablemachine.withgoogle.com/models/W9_qIu14Y/
- ii. Fake vs. true coral snakes model (model 2): <https://teachablemachine.withgoogle.com/models/9vw2M7LJw/>
- iii. Model with variety (model 3): <https://teachablemachine.withgoogle.com/models/Sc8mKQsSo/>

Tip. if you click on the link, the model will open on 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 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 do.

This approach of connecting chatGPT with powerful external tools has been largely explored. For example, [the group behind Matematica](#), a famous numerical software, used this approach as so chatGPT will give educated responses.

2.2.2. Parametrization of openAI APIs

The step by step of the algorithm is on fig. 3. The parametrization on each step is as following.

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 alternative. The AI generated description is triggered just when no description is provided by the user.


```

1 //It will return a promise, with a call to openAI
2 return await this.openai.chat.completions.create({
3   max_tokens:300,//size of reponse, in tokens
4   model: "gpt-4-vision-preview",//model used on the call
5   messages: [//Message sent
6     {
7       role: "user",
8       content: [
9         { type: "text",
10          text:
11            'There is a snake on this image.
12             Describe it.
13             Focus on the colors and patterns.
14             Ex. the snake is black, red, and white.
15             The colors are in strip patterns.
16             The black strips appear more easily than
17             the red ones.' },
18         {
19           type: "image_url",
20           image_url: {
21             "url": this.selectedImage,
22           },
23         },
24       ],
25     },
26   ],
27 }).then((response)=>{
28   //once the promised is finished
29   //getting just the response
30   return response.choices[0].message.content;
31 })

```

```

//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 openAI API is the same as previously. The only change is the parametrization.

```

1  const options:any = {
2  model: "gpt-3.5-turbo-1106",
3  messages: messages,
4  tools: tools,
5  tool_choice: "auto",
6  };

```

```

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 capability, and with a large context (the size of possible inputs to the model, basic chatGPT is about 2 pages).

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

- i. 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 highest probability: name and the probability;
- ii. Fake vs. true coral snakes model (model 2): Should be used only on snakes that looks 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 highest probability: name and the probability;
- iii. 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 highest probability: name and the probability.

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

3rd step: gathering all the response, and getting a conclusion

On this last part, we use again *gpt-3.5-turbo-1106* for creating a final response. On the final information sent to *gpt-3.5-turbo-1106*, we have a JSON response from SnakeFace.

```

1  this.openai.chat.completions.create({
2  model: "gpt-3.5-turbo-1106",
3  messages: messages,
4  }).then((secondResponse)=>{
5
6  this.message_chat.push(
7  'Em conclus o :
8  \n ${secondResponse.choices[0].message.content} ');
9  }); e

```

```

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 it [here](#) the complete method as Gist (a GitHub file of code).

2.3. External resources

In addition to the links already provided, you can also access a version of the model deployed [here](#).



Figure 1. SnakeFace logo.

3. Results and Discussion

At the end of our last paper Pires and Braga (2023), we proposed an app that would connect SnakeFace with the textual capabilities of chatGPT (fig.2).

SnakeFace is a transfer learning based tool created to classify snakes. Different from general computer vision models such as MobileNet, 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 on the image, even adding a label, but bad at actually finding the species. As we are going to see, also the vision model of openAI also overlook Brazilian species. Our goal is making SnakeFace a part of a conversational AI, and still getting it right for native species from Brazil.

If you see with attention, our first idealization of SnakeChat (fig.2) was by using chatGPT for guiding which algorithm should be used on a possible design with small algorithms. We had in mind a manual function calling, now openAI made available an option to automatically do that. Furthermore, since we have 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.

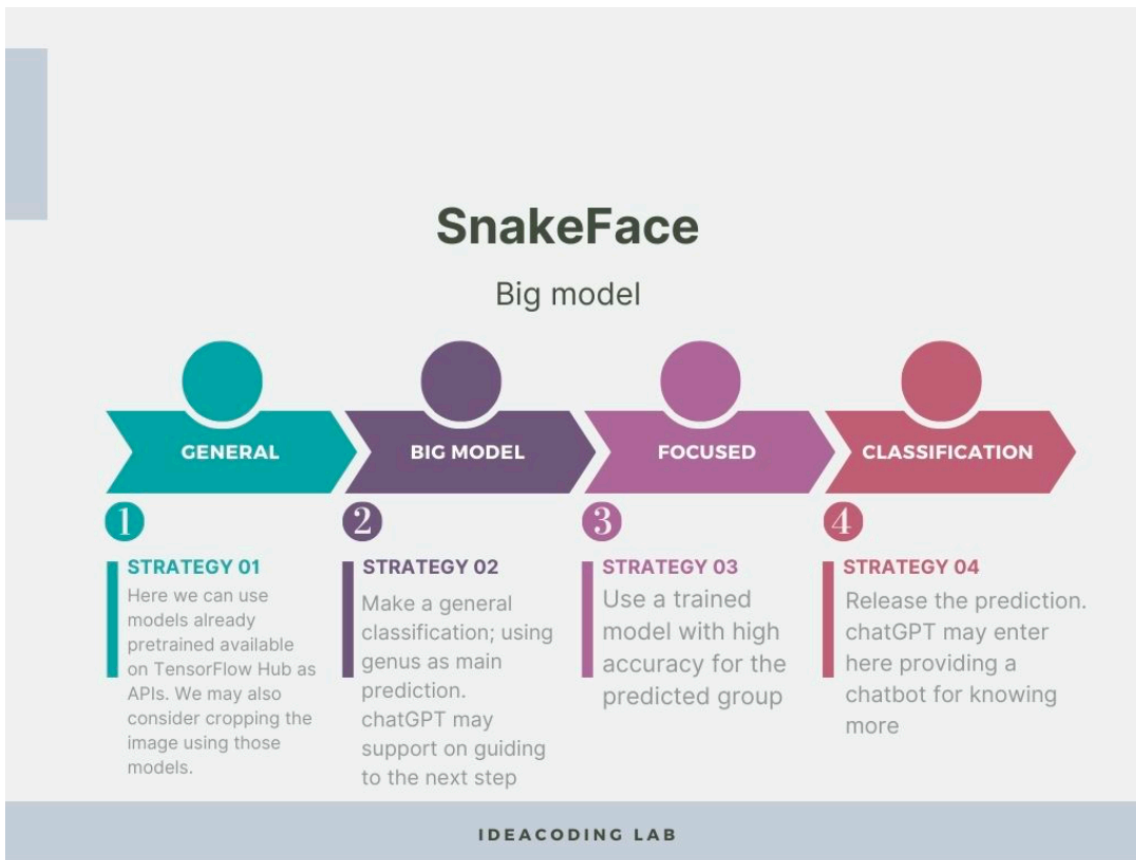


Figure 2. First idealization of snakeChat. Source: Pires and Braga (2023)

One important finding we did on our paper about SnakeFace Pires and Braga (2023)

is that when a sub-model does well on their small group, they also generally do well at a big model. It means that small models may not be a good idea just to improve the prediction: it does not seem to improve just because it is smaller. Small models are interesting due to two reasons mainly: i) when you retrain the model, we do not lose what we did good at the training stage, with a big model, when we retrain, we lose what we did well on 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 to classify badly; one proof of that is that if you remove the background, surprisingly, it does not improve the prediction, it may worsen the prediction.

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 for making our app smarter.

On the following image (fig.3), we have how we executed fig.2 using as pivot the latest APIs from openAI. Different from the initial plan to use just chatGPT API, we could go further thanks to the latest releases from openAI as APIs.

Flowchart for SnakeChat

A conversational AI

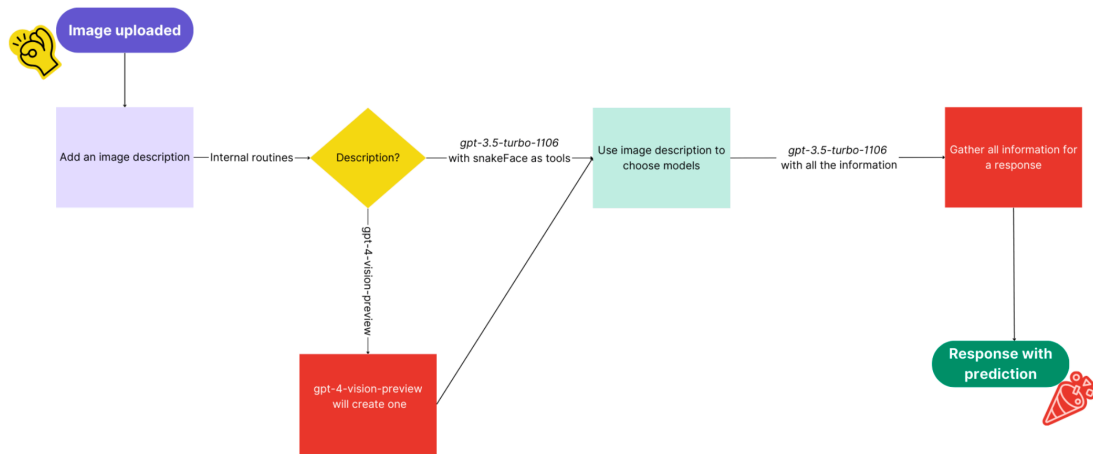


Figure 3. Executed version of SnakeChat based on SnakeFace vision. See that for the last test we present we have replaced *gpt-3.5-turbo-1106* by *gpt-4-1106-preview*.

3.1. Conversations

3.1.1. *Oxyrhopus rhombifer*

We are going to start with what the current app does quite well. Rarely, it will misidentify. First, we need to bring to attention.

- The algorithms from openAIs we are using are stochastic: they offer different responses upon the same input. This means that sometimes the response from openAI API may not be 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 openAI API. This may occasionally lead to a confusing final response;
- We have not yet figure out a good solution, but you can read the probabilities, and pick the highest one, but for some reasons, openAI API will pick not necessarily the highest probability as final response. It can be confusing. It seems to trust more the textual description than the numbers. It seems to improve with their GPT 4 model, see our last conversation;

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

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

One example is when SnakeFace gets it right, but openAI API guesses it wrong. Since we are logging all the responses, someone with the ability to handle those contractions 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, less unnecessary numbers to confuse the final response.

One nice feature from openAI API that may come handy in the future is that they allow to train their textual models, called fine-tuning; and then the model will be available to be called. We mean, we can in the future 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, that could not identify precisely Brazilian snakes. MobileNet was trained for generic objects.

Our input image is below (fig.4): it is a clear image, no visual distractions, simple background.



Figure 4. *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](#).

We are going to test first a human-generated image description.

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

We asked *gpt-3.5-turbo-1106* guessing 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 as chat logs. It guess as *coral snake*, which is wrong. As said before, the behavior of openAI APIs is stochastic, in some tests, they actually guess right.

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

Probabilidade de 82% de ser uma *Oxyrhopus rhombifer* usando SnakeFace, um sistema de identificação por imagens. [correct]

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

Second call, using the generic model:

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

Remember that openAI API decides which model(s) to call: those calls were done automatically by openAI API. It has at its disposal three models, discussed in Pires and Braga (2023); see the method section for minimal details therein.

If we see logs, *gpt-3.5-turbo-1106* decided to call two models we made available from SnakeFace: *true_and_false_coral* and *generic_model_snake*; the former is used to classify false and true coral snakes, it was created to try to separate them from each other, see Pires and Braga (2023); whereas the latter is for generic snakes, 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 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* guess on the snake, whereas the second guess confirmed.

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

The snake in the photo is most likely an *Oxyrhopus 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 15% probability for *Oxyrhopus rhombifer*. The false vs. true coral snake gave lower than 1% 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 human: *gpt-3.5-turbo-1106* on 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 use SnakeFace mistakes to confirm their initial guess).

One may ask ourselves how other researches are doing. We did a small literature survey on our other paper Pires and Braga (2023), we are not going to replicate it herein.

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 available, that we could just plug in to our SnakeChat and test. We are opened for 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.2). This approach makes it easier to compare models, as long as they are easily available for being used.

Now, we are going to test the system without 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 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 description 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 SnakeFace models predicted the correct snake (*Oxyrhopus rhombifer*): 82% and 69% ; one of the models misclassified 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 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 *Calliophis*, *Hemibungarus*, or *Sinomicrurus* genera in the Old World, are known for their distinctive red, black, and yellow/white banding. However, the phrase "*red touch black, venom 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, 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 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 tuning 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 Braga (2023): 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 venomous snake as no-venomous. We

called it error type II. In fact, it was the main focus of another research Felipe Guimarães Dos Santos (2021). 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 image, it is an overall model behaviour. The version we have used now was retrained after the paper, and it fooled the current model.



Figure 5. *Erythrolamprus aesculapii*. Source: Pires and Braga (2023).

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 ones, whereas the black and white ones seems at the same size.

We asked also their latest model to guess, it guess equally wrong compared to the current version we are using from their API. Guessed coral snake. As a result, it decided to use just one model. Guessed as *Micrurus frontalis* (100%); indeed, compared fig.5 and fig.6., 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.



Figure 6. *Micrurus frontalis*; compare with fig.5, they are essentially the same visually.
Source: [Wikipedia](#)

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 right answer corrected by SnakeFace and accepted, but also to what we have already seen: a sort of confirmation bias.

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 high level of intelligence.



Figure 7. *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.

Surprise as it may seem, this description left to guessing as coral snake with the model used so far. We are changing both the function calling algorithm and the final response algorithm (see 2). 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*. In their website, they suggest to fine-tune *gpt-3.5-turbo-1106* for getting 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, it also avoided calling unnecessary functions. When guessing as coral snake, as did the previous algorithm, it will call unnecessary functions, creating noise in the final response: creating space for wrong guessing the snake. On the calls we did with this image, they always called just one model: the general model; different from the previous set up, that seems to be calling unnecessary functions, and creating unnecessary results to 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 a 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.

Best of all, this result tends to repeat itself if you upload the same image. It is good!

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 on our other paper Pires and Braga (2023), 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 done 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 no 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.

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 are 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 on their own documentation](#). [We have also here a comment on their forum](#): it may be released in the upcoming feature 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, it will either make a wrong diagnosis, or refuse to do so. 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 are plenty of literature studying the openAI models applied to diagnosis in medicine, their limitations and strengths, as example, but not exhaustive. Ghosn et al. (2023) compared the GPT 3.5 and GPT 4.0. As we expected, GPT 4 was better at answering questions from an standard examination in radiology, and also explaining the answers.

4. Conclusion

On this article, we have discussed a prototype for a conversational artificial intelligence (CAI) focused on snakes classifications. We have used the APIs from openAI as leverage, as 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 proving a more human-like prediction: instead of just providing a species and probability, we provide 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 is generated by an artificial intelligence. Our simulations show a good integration between SnakeFace and OpenAI APIs, what we have named SnakeChat. Our next step would be: i) adding more species of snakes; ii) adding

more information as so to enhance the predictions; iii) keep testing our model and keeping finding new algorithms that can make our system unique.

One way to add new information to the prediction is using INaturalist: they provide an API that can give back information about snakes, such as if they were ever seen on 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 those information into a final response. One path is providing also 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 the last years, both in computer vision and conversational artificial intelligence. The best advances in AI was becoming public APIs, as so they became bricks for programmers, instead of final tools delivered already designed.

References

- Emsley, R. (2023). Chatgpt: these are not hallucinations – they're fabrications and falsifications, *Schizophrenia* 9(52): 1–2.
- Felipe Guimarães Dos Santos, Emanuel A. Machiaveli, J. M. V. M. (2021). Snake classifier: Aplicativo mobile para classificação de serpentes peçonhentas.
https://www.academia.edu/95702218/Snake_Classifier_Aplicativo_mobile_para_classifica%C3%A7%C3%A3o_de_serpentes_pe%C3%A7onhentas.
- Ghosn, Y., Sardouk, O. E., Jabbour, Y., Jrad, M., Kamareddine, M. H., Abbas, N., Saade, C. and Ghanem, A. A. (2023). Chatgpt 4 versus chatgpt 3.5 on the final frcr part a sample questions. assessing performance and accuracy of explanations, *medRxiv* .
URL: <https://www.medrxiv.org/content/early/2023/09/08/2023.09.06.23295144>.
- Kublik, S. and Saboo, S. (2022). *GPT-3: Building Innovative NLP Products Using Large Language Models*, illustrated edn, O'Reilly Media, Incorporated.
- Pires, J. G. (2023). Do jumento ao einstein chatty: usando chatgpt para criar um assistente virtual inteligente usando a plataforma wix, <https://www.amazon.com.br/Jumento-Einstein-Chatty-assistente-inteligente-ebook/dp/BoC1P811QE>.
- Pires, J. G. and Braga, L. H. D. (2023). Snakeface: a transfer learning based app for snake classification, *bioRxiv* .
URL: <https://www.biorxiv.org/content/early/2023/07/06/2023.06.13.544741>
- Wolfram, S. (2023). What is chatgpt doing . . . and why does it work?, <https://writings.stephenwolfram.com/2023/02/what-is-chatgpt-doing-and-why-does-it-work/>.

Declarations

Funding: No specific funding was received for this work.

Potential competing interests: No potential competing interests to declare.