Satellite Image Cloud Detection via AzureOpenAI Inference

1. Objective

To build a lightweight, user-friendly interface where users can upload a satellite image and instantly receive a label — "Cloudy" or "Clear" — by leveraging a Large Language Model (LLM) deployed on Azure OpenAI. The LLM is prompted to perform visual scene analysis and classification, removing the need for users to set up or maintain conventional deep learning models. This approach enables fast, API-driven cloud detection and simplifies workflows for users without machine learning or programming expertise.

2. Problem Statement

Manual inspection of satellite imagery to identify cloud-covered scenes is time-consuming and does not scale for large datasets. Traditional solutions require deploying or fine-tuning pretrained image classification models, which can be technically challenging for many users. There is a need for a simple web-based application that allows users to upload satellite images and instantly obtain a "Cloudy" or "Clear" classification. By leveraging the image understanding capabilities of modern LLMs via Azure OpenAI, we can bypass complex model management and provide an accessible, code-free cloud detection solution suitable for integration into various industries such as media, agriculture, logistics, and climate analytics.

Your task is to build a minimal application — CLI, script, or web-based (Streamlit) — that:

- Accepts satellite images as input (e.g., .jpg, .png)
- Uses LLM for inference
- Returns an output label: **"Cloudy"** or **"Clear"**
- Optionally, shows the image back with its predicted label

This inference-based approach saves time and enables **instant integration into operational pipelines** for media, agriculture, logistics, and climate analytics teams.

3. Inputs / Shared Artifacts

- Azure OpenAI Resource:
    - API key
    - Endpoint URL
    - Deployment name (to be set as environment variables)
- Sample images:
    - https://www.kaggle.com/datasets/hmendonca/cloud-cover-detection/data

4. Expected Outcome

For each uploaded satellite image, the system should:

- Return a **single label**: either **"Cloudy"** or **"Clear"**
- Optionally, display the **confidence score** (e.g., 92.4% confidence in prediction)
- Visually present the image back to the user with the predicted label shown

Example:

- **Input**: image_23.jpg
- **Output**:
    - **Prediction**: Cloudy

5. Concepts Covered

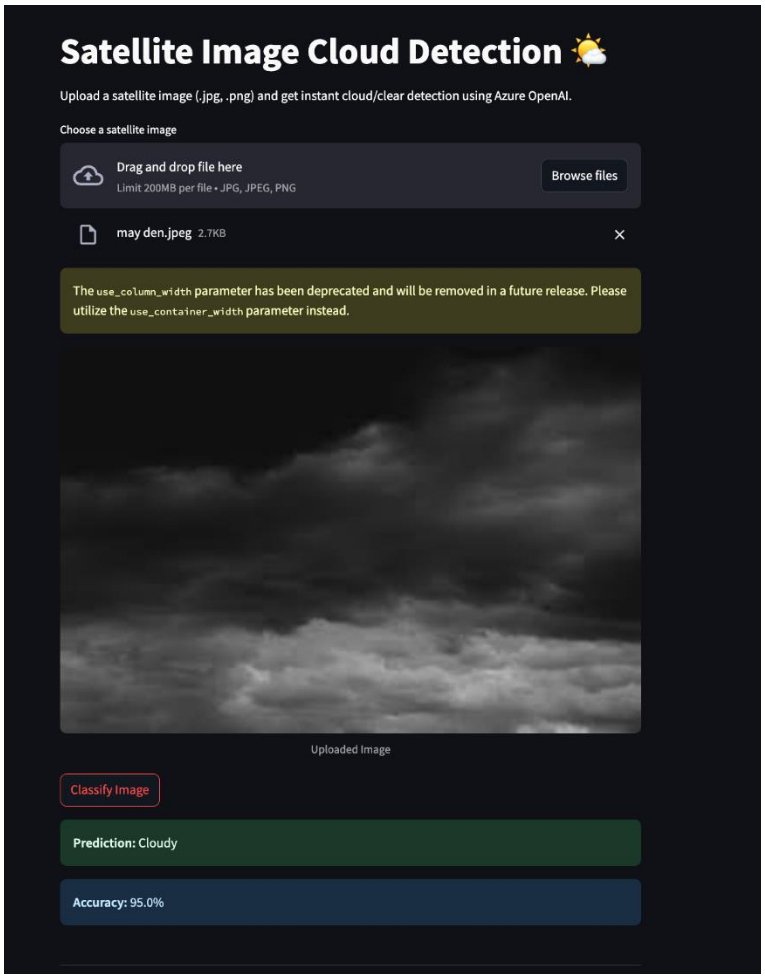
- Azure OpenAI API usage
- Image Classification
- Open-source model
- Pass multimodal data to models

6. Example: Implementation

```
import streamlit as st
import base64
import os
from langchain_openai import AzureChatOpenAI
from pydantic import BaseModel, Field
from PIL import Image
import io

# --- Azure OpenAI Config ---
st.set_page_config(
page_title="Satellite Cloud Detection with Azure OpenAI", page_icon="⛅"
)
st.title("Satellite Image Cloud Detection ⛅")
st.write(
"Upload a satellite image (.jpg, .png) and get instant cloud/clear detection
using Azure OpenAI."
)

# User inputs for secrets/config (can use os.environ for deployment)
azure_endpoint = os.environ.get("AZURE_OPENAI_ENDPOINT")
azure_api_key = os.environ.get("AZURE_OPENAI_API_KEY")
azure_deployment = os.environ.get("AZURE_OPENAI_DEPLOYMENT_NAME")
# --- Model Setup ---
llm = AzureChatOpenAI(
azure_endpoint=azure_endpoint,
azure_deployment=azure_deployment,
api_key=azure_api_key,
api_version="2024-02-15-preview", # Adjust if necessary
)


class WeatherResponse(BaseModel):
accuracy: float = Field(description="The accuracy of the result")
result: str = Field(description="The result of the classification")
```

```
llm_with_structured_output = llm.with_structured_output(WeatherResponse)

# --- File Upload ---
uploaded_file = st.file_uploader(
"Choose a satellite image", type=["jpg", "jpeg", "png"]
)
if uploaded_file:
st.image(uploaded_file, caption="Uploaded Image", use_column_width=True)
image_bytes = uploaded_file.read()
image_data = base64.b64encode(image_bytes).decode("utf-8")
image_pil = Image.open(io.BytesIO(image_bytes)).convert("RGB")

# Prompt
message = [
{
"role": "system",
"content": """Based on the satellite image provided, classify the scene as
either:
'Clear' (no clouds) or 'Cloudy' (with clouds).
Respond with only one word: either 'Clear' or 'Cloudy' and Accuracy. Do not
provide explanations.""",
},
{
"role": "user",
"content": [
{
"type": "text",
"text": "Classify the scene as either: 'Clear' (no clouds) or 'Cloudy' (with
clouds) and Accuracy.",
},
{
"type": "image_url",
"image_url": {
"url": f"data:image/jpeg;base64,{image_data}",},},],},]

# Classify
if st.button("Classify Image"):
```

```
with st.spinner("Classifying..."):
try:
response = llm_with_structured_output.invoke(message)
st.success(f"**Prediction:** {response.result}")
st.info(f"**Accuracy:** {response.accuracy*100:.1f}%")
except Exception as e:
st.error(f"Prediction failed: {e}")

st.markdown("---")
st.caption("Built with Azure OpenAI, Streamlit, and LangChain.")
```



Satellite Image Cloud Detection ⛅

Upload a satellite image (.jpg, .png) and get instant cloud/clear detection using Azure OpenAI.

Choose a satellite image

The `use_column_width` parameter has been deprecated and will be removed in a future release. Please utilize the `use_container_width` parameter instead.

Uploaded Image

Prediction: Cloudy

Accuracy: 95.0%

7. Final Submission

- Complete source code - Implements a Streamlit interface where users can upload satellite images
- Displays the image and the model's response
- **A response log file**

  For each image, include:
    - Filename
    - Model classification (Cloudy / Clear)
    - Confidence (if prompted or inferred)
- **Example (CSV format):**

Filename	**Prediction**	**Confidence**
image_001.jpg	Cloudy	88%