```python
from PIL import Image
from transformers import ViTFeatureExtractor, ViTForImageClassification
import warnings
warnings.filterwarnings('ignore')


#Load the pretrained vision transformer model and faeture extractor
model_name = 'google/vit-base-patch16-224'
feature_extractor = ViTFeatureExtractor.from_pretrained(model_name)
model = ViTForImageClassification.from_pretrained(model_name)

##Path for the image
image_path = '/content/apple.jfif'
image_path = '/content/banana.jfif'
##image_path = '/content/mobile.jfif'
##Load and process the image
image = Image.open(image_path)

inputs = feature_extractor(images = image, return_tensors = "pt")

##perform inference

outputs = model(**inputs)
logits = outputs.logits
predicted_class_idx = logits.argmax(-1).item()
predicted_label = model.config.id2label[predicted_class_idx]

#Extract the name of the food item

food_name = predicted_label.split(',')[0]

#print the food item

print(food_name)
```

```
banana
```

```
import requests
API_KEY = '1xR/oBXk19VVTVOWXFnZOw==OIpHxLRGG8mIB9NZ'
query = food_name
api_url = 'https://api.api-ninjas.com/v1/nutrition?query={}'.format(query)
response = requests.get(api_url, headers={'X-Api-Key': API_KEY})


if response.status_code == requests.codes.ok:
    print(response.text)
else:
    print("Error:", response.status_code, response.text)
```

```
pip install gradio
```

```
Collecting email_validator>=2.0.0 (from fastapi->gradio)
  Downloading email_validator-2.1.1-py3-none-any.whl (30 kB)
Collecting dnspython>=2.0.0 (from email_validator>=2.0.0->fastapi->gradio)
  Downloading dnspython-2.6.1-py3-none-any.whl (307 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 307.7/307.7 kB 29.4 MB/s eta 0:00:00
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair<6.0,>=4
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.10/dist-packages (from jsonsche
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair<6
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair<6.0,>=
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib~=3.
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.10/dist-packages (from rich>=10.11.0->typer<1.
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from rich>=10.11.0->typer<
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio->httpx>=0.24.1->gradio)
Collecting httptools>=0.5.0 (from uvicorn>=0.14.0->gradio)
  Downloading httptools-0.6.1-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.w
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 341.4/341.4 kB 43.9 MB/s eta 0:00:00
Collecting python-dotenv>=0.13 (from uvicorn>=0.14.0->gradio)
  Downloading python_dotenv-1.0.1-py3-none-any.whl (19 kB)
Collecting uvloop!=0.15.0,!=0.15.1,>=0.14.0 (from uvicorn>=0.14.0->gradio)
  Downloading uvloop-0.19.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.4/3.4 MB 95.0 MB/s eta 0:00:00
Collecting watchfiles>=0.13 (from uvicorn>=0.14.0->gradio)
  Downloading watchfiles-0.22.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.2/1.2 MB 85.0 MB/s eta 0:00:00
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->huggingfac
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packages (from markdown-it-py>=2.2.0->rich>=10.11
Building wheels for collected packages: ffmpy
  Building wheel for ffmpy (setup.py) ... done
  Created wheel for ffmpy: filename=ffmpy-0.3.2-py3-none-any.whl size=5584 sha256=80fcc4c29fdf7af86e6997164fcdeebd0f64a852305
  Stored in directory: /root/.cache/pip/wheels/bd/65/9a/671fc6dcde07d4418df0c592f8df512b26d7a0029c2a23dd81
Successfully built ffmpy
Installing collected packages: pydub, ffmpy, websockets, uvloop, ujson, tomlkit, semantic-version, ruff, python-multipart, py
Successfully installed aiofiles-23.2.1 dnspython-2.6.1 email_validator-2.1.1 fastapi-0.111.0 fastapi-cli-0.0.4 ffmpy-0.3.2 gr
```

```
from PIL import Image
from transformers import ViTFeatureExtractor, ViTForImageClassification
import warnings
import requests
import gradio as gr
```

```
warnings.filterwarnings('ignore')

# Load the pre-trained Vision Transformer model and feature extractor
model_name = "google/vit-base-patch16-224"
feature_extractor = ViTFeatureExtractor.from_pretrained(model_name)
model = ViTForImageClassification.from_pretrained(model_name)

# API key for the nutrition information
api_key = '1xR/oBXk19VVTVOWXFnZOw==OIpHxLRGG8mIB9NZ'

def identify_image(image_path):
    """Identify the food item in the image."""
    image = Image.open(image_path)
    inputs = feature_extractor(images=image, return_tensors="pt")
    outputs = model(**inputs)
    logits = outputs.logits
    predicted_class_idx = logits.argmax(-1).item()
    predicted_label = model.config.id2label[predicted_class_idx]
    food_name = predicted_label.split(',')[0]
    return food_name

def get_calories(food_name):
    """Get the calorie information of the identified food item."""
    api_url = 'https://api.api-ninjas.com/v1/nutrition?query={}'.format(food_name)
    response = requests.get(api_url, headers={'X-Api-Key': api_key})
    if response.status_code == requests.codes.ok:
        nutrition_info = response.json()
    else:
        nutrition_info = {"Error": response.status_code, "Message": response.text}
    return nutrition_info

def format_nutrition_info(nutrition_info):
    """Format the nutritional information into an HTML table."""
    if "Error" in nutrition_info:
        return f"Error: {nutrition_info['Error']} - {nutrition_info['Message']}"

    if len(nutrition_info) == 0:
        return "No nutritional information found."
```

```
            return "No nutritional information found."

        nutrition_data = nutrition_info[0]
        table = f"""
        <table border="1" style="width: 100%; border-collapse: collapse;">
            <tr><th colspan="4" style="text-align: center;"><b>Nutrition Facts</b></th></tr>
            <tr><td colspan="4" style="text-align: center;"><b>Food Name: {nutrition_data['name']}</b></td></tr>
            <tr>
                <td style="text-align: left;"><b>Calories</b></td><td style="text-align: right;">{nutrition_data['calories']}
                <td style="text-align: left;"><b>Serving Size (g)</b></td><td style="text-align: right;">{nutrition_data['ser
            </tr>
            <tr>
                <td style="text-align: left;"><b>Total Fat (g)</b></td><td style="text-align: right;">{nutrition_data['fat_to
                <td style="text-align: left;"><b>Saturated Fat (g)</b></td><td style="text-align: right;">{nutrition_data['fa
            </tr>
            <tr>
                <td style="text-align: left;"><b>Protein (g)</b></td><td style="text-align: right;">{nutrition_data['protein_
                <td style="text-align: left;"><b>Sodium (mg)</b></td><td style="text-align: right;">{nutrition_data['sodium_m
            </tr>
            <tr>
                <td style="text-align: left;"><b>Potassium (mg)</b></td><td style="text-align: right;">{nutrition_data['potas
                <td style="text-align: left;"><b>Cholesterol (mg)</b></td><td style="text-align: right;">{nutrition_data['cho
            </tr>
            <tr>
                <td style="text-align: left;"><b>Total Carbohydrates (g)</b></td><td style="text-align: right;">{nutrition_da
                <td style="text-align: left;"><b>Fiber (g)</b></td><td style="text-align: right;">{nutrition_data['fiber_g']}
            </tr>
            <tr>
                <td style="text-align: left;"><b>Sugar (g)</b></td><td style="text-align: right;">{nutrition_data['sugar_g']}
                <td></td><td></td>
            </tr>
        </table>
        """
        return table

    def main_process(image_path):
        """Identify the food item and fetch its calorie information."""
        food_name = identify_image(image_path)
        nutrition_info = get_calories(food_name)
        formatted_nutrition_info = format_nutrition_info(nutrition_info)
        return formatted_nutrition_info

    # Define the Gradio interface
    def gradio_interface(image):
        formatted_nutrition_info = main_process(image)
        return formatted_nutrition_info

    # Create the Gradio UI
    iface = gr.Interface(
        fn=gradio_interface,
        inputs=gr.Image(type="filepath"),
        outputs="html",
        title="Food Identification and Nutrition Info",
        description="Upload an image of food to get nutritional information.",
        allow_flagging="never"  # Disable flagging
    )

    # Launch the Gradio app
    if __name__ == "__main__":
        iface.launch()
```

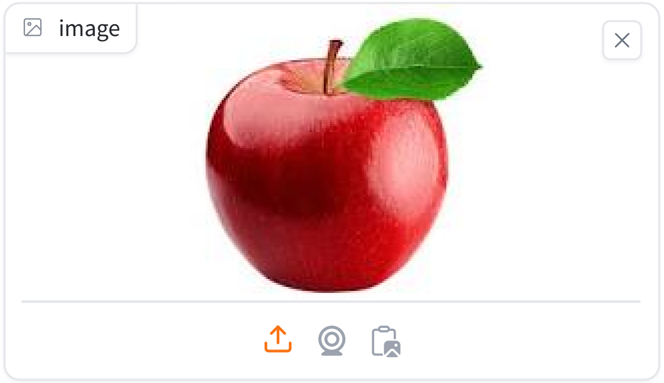
```
Setting queue=True in a Colab notebook requires sharing enabled. Setting `share=True` (you can turn this off by setting `share=

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
Running on public URL: https://960afea2a70278d596.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from Terminal to deploy t
```

Food Identification and Nutrition Info

Upload an image of food to get nutritional information.



Clear

Submit

Nutrition Facts			
Food Name: granny smith			
Calories	58.5	Serving Size (g)	100.0
Total Fat (g)	0.2	Saturated Fat (g)	0.0
Protein (g)	0.4	Sodium (mg)	1
Potassium (mg)	12	Cholesterol (mg)	0
Total Carbohydrates (g)	13.4	Fiber (g)	2.8