Documentation for Image Module

This module implements an image classification program that integrates a pre-trained machine learning model with a RabbitMQ message broker to classify images received via messaging queues. The module performs tasks such as image preprocessing, classification, message handling, and error reporting.

Modules and Libraries

- **os, io**: Used for file path handling and byte stream management.
- **torch**: Provides support for machine learning operations, including inference and tensor manipulations.
- **torch.nn.functional**: Contains the SoftMax function for probability calculation.
- **PIL (Pillow)**: Handles image processing.
- **pika**: Facilitates interaction with RabbitMQ for message publishing and consuming.
- **BSON**: Serializes and deserializes messages to/from BSON format for efficient data handling.
- **transformers**: Provides the AutoModelForImageClassification and AutoImageProcessor for machine learning tasks.

Functions

classify_image (image_data, file_name)

- **Purpose**: Classifies an image using a pre-trained machine learning model and returns the predicted class and confidence score.
- **Parameters**:
    - image_data (bytes): Byte data of the image to be classified.
    - file_name (str): Name of the image file (used in result messages).
- **Returns**:
    - predicted_class (str): The predicted label for the image.
    - confidence_score (float): The confidence score of the prediction.
- **Key Steps**:

1. Opens the image from byte data.
2. Preprocesses the image using the model's preprocessor.
3. Performs inference using the model.
4. Computes confidence scores using the SoftMax function.
5. Prints and returns the classification results.

---

publish_to_rabbitmq (routing_key, message)

- **Purpose**: Publishes a message to a specified RabbitMQ queue.
- **Parameters**:
    - routing_key (str): Name of the RabbitMQ queue.
    - message (dict): Data to be published, serialized to BSON format.
- **Key Steps**:

1. Connects to the RabbitMQ server.
2. Ensure the specified queue exists.
3. Publishes the BSON-encoded message to the queue.

---

on_message_received (ch, method, properties, body)

- **Purpose**: Handles incoming messages, performs image classification, and publishes results.
- **Parameters**:
    - ch, method, properties: RabbitMQ message metadata (unused but required by pika callback).
    - body (bytes): BSON-encoded message payload containing image data and metadata.
- **Key Steps**:

1. Decodes the incoming BSON message.
2. Extracts image data and metadata.
3. Classifies the image using classify_image

4. Publishes the classification results or error messages to RabbitMQ.

---

consumer_connection (routing_key)

- **Purpose**: Creates a RabbitMQ consumer that listens to a specified queue for incoming messages.
- **Parameters**:
    - routing_key (str): Name of the RabbitMQ queue to consume messages from.
- **Key Steps**:

1. Connects to the RabbitMQ server and declares the queue.
2. Starts consuming messages and processes them using on_message_received.
3. Allows graceful shutdown on keyboard interrupt.

Execution Flow

1. **Model Initialization**:
    - Loads the Microsoft/resnet-50 pre-trained model and its corresponding image processor.
2. **RabbitMQ Consumer Setup**:
    - The script connects to RabbitMQ and listens to the Image queue for incoming tasks.
3. **Message Processing**:
    - When a message is received, it extracts image data, performs classification, and publishes the results.
4. **Error Handling**:
    - Captures exceptions during classification and publishes detailed error messages for tracking.
5. **Entry Point**:
    - The script runs the consumer_connection function to start consuming messages from the Image queue when executed.

Key Features

- **Machine Learning Integration**:
    - Uses a pre-trained ResNet-50 model for high-accuracy image classification.
- **RabbitMQ Messaging**:
    - Handles asynchronous task communication using durable message queues.
- **Error Handling**:
    - Publishes failure messages with detailed information to a separate status queue.
- **Serialization**:
    - Employs BSON for efficient message encoding and decoding.
- **Scalability**:
    - Designed for deployment in scalable environments, such as Docker containers.

Usage

- Ensure RabbitMQ is running locally (localhost).
- Place the script in an environment with Python dependencies installed (torch, transformers, pika, BSON, Pillow).
- Run the script to start taking messages from the Image queue:
    - python image_classification_module.py